Task/Subtask IV01.7
CDRL Sequence A011-001F
25 July 1994

1

SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS) PROGRAM

AF/STARS Demonstration Project Experience Report Version 1.1

Contract No. F19628-93-C-0129
Task IV01 – Megaprogramming Demonstration Projects

Prepared for:

Electronic Systems Center
Air Force Materiel Command, USAF
Hanscom AFB, MA 01731-2116

Prepared Jointly by:

Loral Federal Systems
700 North Frederick Avenue
Gaithersburg, MD 20879

107

and

SWSC/SMX
130 West Paine Steet
Peterson AFB, CO 80914-2320

CLEARED FOR PUBLIC RELEASE, DISTRIBUTION IS UNLIMITED

Task/Subtask IV01.7
CDRL Sequence A011–001F
25 July 1994

SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS) PROGRAM

AF/STARS Demonstration Project Experience Report Version 1.1

Contract No. F19628–93–C–0129
Task IV01 – Megaprogramming Demonstration Projects

Prepared for:

Electronic Systems Center
Air Force Materiel Command, USAF
Hanscom AFB, MA 01731–2116

Prepared Jointly by:

Loral Federal Systems
700 North Frederick Avenue
Gaithersburg, MD 20879

and

SWSC/SMX
130 West Paine Steet
Peterson AFB, CO 80914–2320

REPORT DOCUMENTION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*

2. REPORT DATE
July 25, 1994

3. REPORT TYPE AND DATES COVERED
Final - Version 1.1

4. TITLE AND SUBTITLE
Air Force/STARS Demonstration Project Experience Report

5. FUNDING NUMBERS
F19628-93-C-0129

6. AUTHOR(S)
Lynn Underhill, Loral Federal Systems

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Loral Federal Systems
700 N. Frederick Avenue
Gaithersburg, MD 20879

SWSC/SMX
130 W. Paine Street
Peterson AFB, CO 80914-2320

8. PERFORMING ORGANIZATION REPORT NUMBER
A011-001F

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Electronic Systems Center/ENS
Air Force Materiel Command, USAF
5 Eglin Street, Building 1704
Hanscom Air Force Base, MA 01731-2116

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
N/A

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Statement A - Approved for Public Release, Distribution is Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This Experience Report is a product of the Space Command and Control Architectural Infrastructure (SCAI) Air Force/STARS Demonstration Project. Its purpose is to present the lessons learned in the course of applying Software Technology for Adaptable, Reliable Systems (STARS) megaprogramming technologies to the SCAI Demonstration Project. Specifically, the report will present the lessons learned in applying the formal STARS process driven, technology supported, domain-specific reuse technologies to:

1) Implement an operational space control capability for the Commander in Chief (CINC) Mobile Alternate Headquarters (CMAH).

2) Further the establishment of a product-line organization with the Air Force Space and Warning Systems Center (SCAI).

14. SUBJECT TERMS
Lessons Learned, Experience, Technology Transition

15. NUMBER OF PAGES
106

16. PRICE CODE
N/A

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

AF/STARS
SCAI

Space Command and Control Architectural Infrastructure (SCAI)

Air Force / STARS Demonstration Project

Experience Report

Version 1.1 - 25 July 1994
Prepared Jointly by:

Loral Federal Systems - Gaithersburg
700 North Frederick Avenue
Gaithersburg, MD 20879

and

SWSC/SMX
130 W. Paine Street
Peterson AFB, CO 80914-2320

Preface

This document was developed by the Loral Federal Systems-Gaithersburg, located at 700 North Fredrick Avenue, Gaithersburg, MD 20879 and SWSC/SMX, 130 West Paine Street, Peterson AFB, CO 80914-2320. Questions or comments should be directed to Captain Clint Heintzelman at 719-554-6533 (cheintze@spacecom.af.mil) or Ms. Lynn Underhill at 719-554-6579 (underhl@wmavm7.lfs.loral.com).

Contributions to this document have been provided by the following organizations: Air Force Space Warning Systems Center, CACI, ccPE, Kaman Sciences Corp, Loral Federal Systems, PRC, Rational, SEI, SET, and TRW.

Forward

Purpose

This Experience Report is a product of the Space Command and Control Architectural Infrastructure (SCAI) Air Force/STARS Demonstration Project. Its purpose is to present the lessons learned in the course of applying Software Technology for Adaptable Reliable Systems (STARS) megaprogramming technologies to the SCAI Demonstration Project. Specifically the report will present the lessons learned in applying the formal STARS process driven, technology supported domain-specific reuse technologies to:

(1) Further the establishment of a product-line organization within the Air Force Space and Warning Systems Center (SWSC).

(2) Implement an operational space control capability for the Commander In Chief (CINC) Mobile Alternate Headquarters (CMAH).

Audience

This document is targeted to a number of different audiences.

(1) Advanced Research Project Agency (ARPA)

(2) Space and Warning Systems Center (SWSC)

(3) Air Force Space Command

(4) Air Force Materiel Command

(5) Department of Defense Community

(6) Software Engineering Community

Assumptions

This document is written with the following assumptions about its audience:

(1) The reader has a working knowledge of software engineering practices, management, development and evolution.

(2) The reader has a working knowledge or an interest in gaining an understanding of the support elements that make up the STARS megaprogramming paradigm: process-driven development, domain-specific reuse, and Software Engineering Environments.

(3) This version of the Experience Report is providing experiences and lessons learned during the Preparation Phase, the first of three project phases. Experience Reports will be published during each project phase.

Document Structure

This document is organized into 6 chapters. A description of the contents of each chapter is provided in the following table (see Table 1)

Chapter	Title	Description
1	Overview	Provides a summary of the project history, vision, mission, goals, management structure, technical approach and schedule.
2	Domain Engineering/ Application Engineering	Summarizes the project's experience in integrating STARS technologies with incumbent approaches to develop engineering processes that address the scope of both Domain and Application Engineering. Summarizes the project's experience in testing the applicability of expanding the C^2AI to the space domain.
3	Process Support	Summarizes the project's experience in preparing the organization to be process-driven and in developing the SCAI Process Architecture.
4	SEE Support	Summarizes the project's experience in integrating and using advanced software tools into a Software Engineering Environment (SEE).
5	Metrics	Summarizes the project's experience in developing a metrics strategy, and initializing the metrics program.
6	Conclusion	Final Remarks

TABLE 1. Document Organization

Table of Contents

List of Figures

List of Tables

Executive Summary

The Space Command and Control Architectural Infrastructure (SCAI) Demonstration Project was selected as the Air Force Software Technology for Adaptable, Reliable Systems (STARS) Demonstration Project and paired with Loral Federal Systems, Gaithersburg, MD. The project is administrated by the Air Force Space Command (AFSPC) Space and Warning Systems Center (SWSC). It is a three year program which is being conducted from October 1992 through October 1995. The partnership was chartered by a Memorandum of Agreement (MOA) signed between the Air Force and ARPA.

The MOA defines the scope of the Demonstration Project as a joint project to apply the STARS megaprogramming paradigm to the current space mission. Megaprogramming is defined by the STARS program as "process-driven, domain-specific reuse based, technology supported" software development. Personnel from LORAL Federal Systems, one of the STARS prime contractors, are partners with the SCAI team. CACI, ccPE, Kaman Sciences Corp, PRC, Rational, Robbins Gioia, SET, and TRW perform various contractor roles on the project.

Air Force Space Command is responsible for the maintenance of Cheyenne Mountain Command and Control Systems and is interested in using new software technologies to realize lower maintenance costs and improve quality and reliability. The AFSPC SWSC has the mission to:

(1) Support combat operations by developing and maintaining mission-critical software to meet the requirements of North American Aerospace Defense Command (NORAD), United States Space Command (USSPACECOM) and AFSPC command, control and intelligence centers.

(2) Ensure the operational and technical integrity of national attack warning and space control systems and provide space and warning software engineering and configuration management.

The STARS megaprogramming and Reusable Integrated Command Center (RICC) architecture infrastructure technologies are to be demonstrated by using them to develop an operational space capability for the Commander in Chief (CINC) Mobile Alternate Headquarters (CMAH). Following the success of the SCAI project the technologies will be transitioned for use throughout the SWSC organization.

AFSPC is applying new software technologies, like Open Systems Environments, to realize lower maintenance costs and improve quality and reliability. The Reusable Integrated Command Center (RICC) technology, developed for the SWSC by TRW under the Air Force Embedded Computer Resource Support Improvement Program (ESIP), is the first set of artifacts to result from the SWSC effort to implement a common architectural approach. This technology includes tools for generating Ada code, application definition files, and reusable domain components.

The SCAI project has the following objectives:

(1) Apply the Reusable Integrated Command Center (RICC) technology to SCAI development and demonstrate its benefit.

(2) Institute cooperating Domain Engineering and Application Engineering (DE/AE) processes including those for ongoing process improvement.

(3) Instantiate a process driven Software Engineering Environment to support megaprogramming.

(4) Demonstrate that the SCAI application was built "cheaper, better, and faster" using the new process.

The Domain Engineering life-cycle supports the establishment of a product-line within the SCAI domain. Domain Engineering is defined as the analysis performed across a family of systems to determine the commonality and variability within the domain. The systems' common assets are represented in models and architecture, which provide the framework for developing reusable assets, and for reusing those assets across the product-line during Application Engineering.

The Application Engineering life-cycle is defined by the engineering activities required to develop the space control software for the CMAH. The capability being implemented will be limited to the functionality needed to:

(1) maintain the satellite catalog to a level of accuracy sufficient for space surveillance sensors to acquire observations on tasked satellites,

(2) process major space surveillance events, and

(3) process major space threat events.

The functions are modeled after identical functions currently incorporated in Space Defense Operations Center (SPADOC) 4C, V1, and planned for SPADOC 4C, V2. The operator interface to these functions is to have the look and feel of SPADOC 4C, although user approved deviations are permitted in order to adhere to austerity guidelines, to comply with open systems standards such as X Windows, or when feasible, to provide enhancements.

Metrics are being collected and are tied to the project goals through critical success factors identified for each goal and are designed to give project members visibility into how well the project is doing in attaining the goals. To-date, effort (man-hours/work breakdown structure task) is the primary metric being collected. However, software tool usage, SEE utilization and schedule/resource prediction features are all targets of interest for collection during the Performance Phase.

This version of the Experience Report provides project background information, the project experiences and lessons learned for each of the megaprogramming focus areas, during the Preparation Phase of the Demonstration Project. Subsequent Experience Reports will be delivered yearly and will provide the experiences and lessons learned during the Performance and Reflection Phases of the project.

The project has made significant progress in the following areas:

(1) Development of an Application Engineering Process and project consensus on that process.

(2) Development of project consensus on the Architectural Framework and the domain/application modeling artifacts.

(3) Demonstration that the Architectural Infrastructure is applicable to other C^2 domains.

The **primary lessons-learned** during the Preparation Phase (October 1992-October 1993) are:

(1) The SCAI team has elaborated the STARS Two Life-Cycle Model to show very close interaction between Domain Engineering and Application Engineering, in order to constantly validate the domain models against real applications in the domain. The models must be developed iteratively to avoid the creation of complete but invalidated models. In fact, the team has developed a working hypothesis that application level models, to a large extent, should be views of domain level models.

(2) Architectural layering in the SWSC domain provides good guidance for forming the functional organizations needed to support the product-line (hypothesis partially validated by experience to date).

(3) The Plan/Enact/Learn paradigm, elaborated in the STARS Conceptual Framework For Reuse Processes (CFRP), applies to all major project activities, including the formulation of the project's approaches and processes as well as the development of the domain and appli-

cation products. Since an organization seeking to transition to megaprogramming can anticipate a significant amount of technology transition and integration, it should consciously plan an incremental build up of its approach.

(4) How technology is transitioned between organizations is as important as the technologies that are transitioned. This project spent a lot of time becoming familiar with all of the technologies prior to being able to integrate them into an approach. This was true for developing our approach for defining project processes as well as for defining our Domain/Application Engineering approach.

(5) Acquisition and integration for the SEE should be planned and developed in step with the incremental development of the process.

(6) Metrics definitions can be used to help the project focus on goals and success indicators early in project life-cycle. We successfully adjusted the Goal/Question/Metrics (GQM) approach to construe the questions as success indicators. Small focus groups of 2 to 5 people were an efficient way to refine goals, subgoals and success indicators to express the projects focus.

The remainder of the Experience Report provides an overview of the SCAI project approach and the experience gained during the Preparation Phase (October 1992 through October 1993). The report will be updated at the end of 1994, mid-way through the project, and in early 1996, at the completion of the project.

1.0 Overview

The purpose of this report is to present the experience and lessons-learned in the course of applying STARS megaprogramming technologies to the SCAI Demonstration Project. The report covers experiences related to Domain Engineering/Application Engineering, Process Support, Software Engineering Environment Support, and Metrics. For each of the above areas the report provides a brief description and overview, followed by plans, summary of accomplishments, lessons-learned, and recommendations.

This section provides introductory material that provides context for the experience sections that comprise the remainder of the report. The following topics are covered:

(1) Background - provides a brief discussion of the STARS program, the Air Force organization paired with STARS for the Demonstration Project, and the nature of the project itself;

(2) Vision, Mission, and Goals - provides the long-range and short-term motivations for the project;

(3) Project Organization - discusses the Air Force organization responsible for the Demonstration Project, as well as the relevance of the project to several outside organizations; and

(4) Overview of Technical Approach - introduces the technologies being applied to the project, and the approach being used to combine them into a practical megaprogramming process.

1.1 Background

This subsection provides brief discussions of the STARS program, the Air Force organization paired with STARS for the Demonstration Project, and the nature of the project itself.

1.1.1 The STARS Program

The Advanced Research Project Agency (ARPA) STARS Prime program was initiated in 1988 to create cooperative development work from a very broad industry experience base, reduce risk, and accelerate acceptance of changing technology. The contract was awarded to three leading defense systems integrators, Boeing Aerospace Company, IBM Federal Systems Division (now Loral Federal Systems), and Unisys Defense Systems.

The current STARS program is a technology development, integration and transition program to demonstrate a process-driven, domain specific, reuse-based approach to software engineering that is supported by appropriate tool and environment technology. This approach is often referred to as "megaprogramming".

Megaprogramming is a product-line (family of systems) approach to the creation and maintenance of software intensive systems. It is characterized by the reuse of software life-cycle assets within a product-line including common architecture and components. Megaprogramming also includes the definition and enactment of disciplined processes for the development of applications within the product-line and for the development and evolution of the product-line itself. [1]

A mission of the STARS program is to accelerate the transition to the megaprogramming paradigm. To accomplish this, STARS is jointly sponsoring Demonstration Projects with each of the three Services. The major objectives for each of the projects are to:

(1) Apply a megaprogramming paradigm to the development of software for an actual DoD system. The system will be produced using new approaches, and will establish the credibility of the megaprogramming approach.

1. Abstracted from a paper titled *STARS Program History 1983-1993* Version 1.1 assembled by Joel Trimble

(2) Make quantitative and qualitative measures of the effects of megaprogramming on the development effort and the resulting product. Produce reports documenting lessons learned in applying megaprogramming that will help others as they begin using megaprogramming and provide feedback on how tools and processes worked in practice.

(3) Transition to the Demonstration Project organization the capability to practice megaprogramming. When this demonstration is over, the Demonstration Project organization will be able to continue using megaprogramming on this project and should be able to apply megaprogramming to other projects without assistance from STARS.

1.1.2 The Air Force Space Command / Space and Warning Systems Center

Air Force Space Command (AFSPC) is responsible for the maintenance of Cheyenne Mountain Command and Control Systems (see Figure 1) and is interested in applying new software technologies to realize lower maintenance costs and improve quality and reliability. The AFSPC Space and Warning Systems Center (SWSC) has the mission to:

(1) Support combat operations by developing and maintaining mission-critical software to meet the requirements of North American Aerospace Defense Command (NORAD), United States Space Command (USSPACECOM) and AFSPC command, control and intelligence centers.

(2) Ensure the operational and technical integrity of national attack warning and space control systems and provide space and warning software engineering and configuration management.

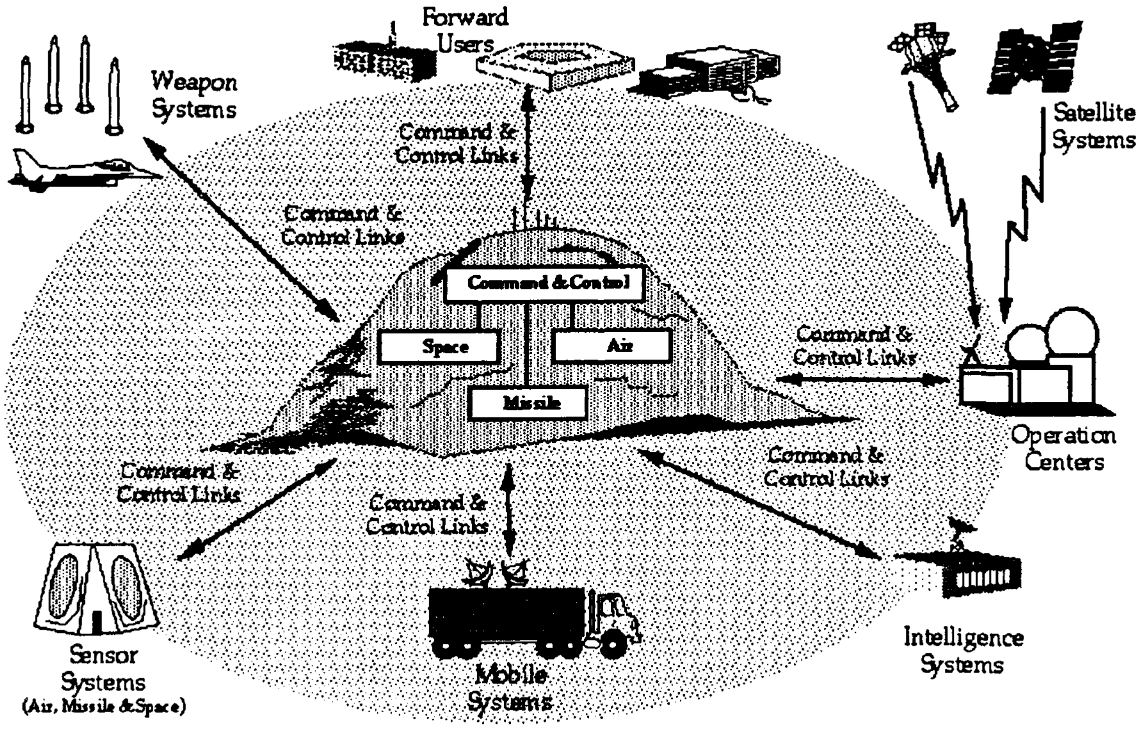


Figure 1. Current Systems of CMAFB

The AFSPC legacy predates the recognition of the importance of a disciplined software engineering process. Stovepipe[1] implementations have precluded reuse, diverse tools and methods have precluded

1. No intent to share architecture or information except at interfaces.

resource sharing and system-oriented organizations have precluded the transfer of technologies. AFSPC currently maintains in excess of 10 million lines of code on 27 separate operational systems written in 24 languages.

In 1990 the AFSPC SWSC initiated an effort to develop a strategy to move to an architecture based on Open Systems Environments as quickly as possible. While the SWSC usually procures systems through Electronic Systems Center (ESC), in some cases, on an experimental basis, the SWSC has attempted to perform some new development itself. As a result of one of these experiments, the SWSC, in cooperation with TRW, set about to demonstrate large scale reuse. They developed a set of architectural components, called the Command and Control Architecture Infrastructure (C^2AI), that promises to significantly reduce systems development time and cost while increasing quality.

The concept for the C^2AI was developed in the SWSC as a result of analyzing the systems that the SWSC is responsible for. Leveraging work accomplished by TRW on a production contract called CCPDS-R (Command Center Processing and Display System Replacement), the SWSC contracted TRW to advance CCPDS-R techniques and demonstrate a subset of the Cheyenne Mountain Missile Warning requirements, on a pilot program called the Reusable Integrated Command Center (RICC).

1.1.3 The AF/STARS Demonstration Project

The Space Command and Control Architectural Infrastructure (SCAI) Demonstration Project was selected as the Air Force STARS Demonstration Project and paired with Loral Federal Systems. The project is administrated by the Air Force Space Command (AFSPC) Space and Warning Systems Center (SWSC). It is a three year program which is being conducted from October 1992 through October 1995.

The Air Force selected the SCAI Demonstration Project to be their STARS Demonstration Project. The progress the SWSC had made in the analysis of the C^2 domain, the definition of the architectural infrastructure and the infrastructure's promise to significantly reduce systems development time and cost made the SWSC an ideal partner. Not only had the SWSC begun megaprogramming work, they were ready and interested in developing quantitative and qualitative measures of the effects of that work. The SWSC expects to profit from its partnership with STARS by accelerating its shift to megaprogramming, using STARS technologies in the following areas:

(1) Reuse-based engineering - instituting a product-line "Domain Engineering" process that focuses on identifying and exploiting the commonality in the domain family of interest.

(2) Process-driven development - planning, and enacting the software development and maintenance process in a disciplined, repeatable way.

(3) Technology-supported process - providing an open and extendable Software Engineering Environment that provides an advanced level of integrated support for the process.

The purpose of the SCAI Demonstration Project is to apply the STARS megaprogramming software engineering approach and the C^2AI to the current space mission. The space algorithms written in FORTRAN will be re-engineered into Ada components and existing AFSPC Command and Control Architectural Infrastructure components will be evolved and integrated into a new space operations control capability for the Commander In Chief (CINC) Mobile Alternate Headquarters (CMAH). The capability to be implemented will be limited to the functionality needed to:

(1) Maintain the satellite catalog to a level of accuracy sufficient for space surveillance sensors to acquire observations on tasked satellites,

(2) Process major space surveillance events, and

(3) Process major space threat events.

The functions will be modeled after identical functions currently incorporated in Space Defense Operations Center (SPADOC) 4C, V1, and planned for SPADOC 4C, V2. The operator interface to these functions will have the look and feel of SPADOC 4C, although user approved deviations will be permitted in order to adhere to austerity guidelines, to comply with open systems standards such as X Windows, or when feasible, to provide enhancements.

SCAI will be capable of rapidly accepting and storing satellite catalog and sensor observation data from SPADOC and, in turn, providing this data to SPADOC. SCAI will run in two modes, an on-line mode and an off-line mode. In the on-line mode, SCAI will maintain the catalog autonomously, and will process both major space surveillance and space threat events. In the off-line mode, SCAI will accept the SPADOC 4C database and update its own database with the current element sets and observations at user-specified time intervals. No observations or events will be processed in the off-line mode.

The products and applications developed in the SCAI effort will follow an evolutionary path from the STARS Demonstration Project to a fully deployable system for the United States Space Command (USSPACECOM), with reusable components capable of satisfying other Command and Control missions.

The SCAI System will be integrated into the Mobile Command and Control System (MCCS) development program.The mission of the MCCS is to provide automated Command, Control, and Communications (C^3) for the NORAD and USSPACECOM Mobile Consolidated Command Center (MCCC). In the operational environment, the MCCC vans are geographically dispersed so that essential command and control functions are available should communications with the Cheyenne Mountain Complex (CMC) be lost. Under the Mission Support Segment (MSS) of the MCCS program, Space Control, Air Defense, and Missile Warning missions will be combined into an integrated C^3 system, using STARS development technologies to their fullest potential.

1.2 Vision, Mission and Goals

The SCAI team has evolved the following statements of vision and intent associated with the Demonstration Project.

SWSC Vision in Connection with the Demonstration Project:

> We are a highly productive, thriving engineering team, using megaprogramming to provide mission-effective systems in our product-line, and delighting our customers with our quality and responsiveness.

SMX Mission during the Demonstration Project Time Frame:

> Formalize a practical megaprogramming process that combines the best of the STARS and RICC technologies, proving its viability by building and deploying an operational system in the space and warning domain and initiating the transition of the process and technology throughout the SWSC.

SCAI Demonstration Project Goals:

The goals for the Demonstration Project are based on the above vision and mission statements and are a joint AF/STARS work product. The team considers these goals to be central to the project's strategy. They provide the motivational basis for all key project activities: notably, the project planning, the process definition and enactment work, and the metrics activity.

Table 8 presents a top level view of the SCAI goals from a perspective that is consistent across all three STARS Demonstration Projects.

Capability	Current Baseline	SCAI Goals *Product/Approach*	Organizational Vision *Institutionalization*
Space Software Baseline	FORTRAN, JOVIAL, Assembler. (3.5M LOC)	Re-engineered in Ada using system architecture.	Multiple space systems based on common models and reusable assets.
Software Process (SWSC)	Defined high level processes. Repeatable at lower levels.	Partial process-driven, reuse-based capability	Managed with continuous process improvement.
Software Reuse (SWSC)	Prototype architectural infrastructure with code generators (UNAS & RICC).	Domain requirements and architectural models, domain library, & reusable assets.	Product-line process with managed reuse.
Software Architecture (SWSC)	System-specific/evolution limited	Evolution enabled	SWSC-wide architectural strategy.
Software Engineering Environment (SWSC)	System-specific	Partial enactment with integrated processes	Adaptable and open. SWSC SEE strategy, integrated assets
People Skills (SWSC)	Diverse, system-specific	Trained in SCAI technologies	Focused on Technologies for Product-Line

TABLE 1. STARS View of AF/STARS Demonstration Project

This high-level depiction summarizes the Demonstration Project objectives according to a set of categories that were developed cooperatively by all three Demonstration Projects. Interested readers may wish to refer to the other two projects' experience reports to compare the ways the three projects are interpreting these common categories.

The information in the table was derived from a much more detailed set of goals that were developed by the SCAI team during the Preparation Phase. The SCAI goals are mapped into three major groupings: product goals, approach goals, and institutionalization goals. The rightmost two columns indicate how these goal groupings map to the information in the table: the SCAI Goals column was derived from the more detailed product and approach goals adopted by the SCAI team, and the Organizational Vision column was derived from the more detailed institutionalization goals.

The following paragraphs provide a synopsis of the specific SCAI goals.

(1) Product Goals

Theme: Build a Real System

- Develop a space mission capability suitable for operational deployment by October, 1995.

  The system must fulfill a usable subset of existing CMC Space Mission capability, be approved by users, meet approved QA/CM requirements, and realize key product quality characteristics. It is assumed that there will be no extensive training of users required (no more than a 1/2 day for experienced users), and that the users will have the opportunity to accept the system incrementally as it is developed. The system is expected to be maintainable, new capabilities added easily, and operator productivity should be as good or better than with current systems.

- Develop reusable domain assets.

  The development of the space mission capability must produce a legacy of reusable assets to include an architecture, models, components and processes. These artifacts should be documented, catalogued, understandable and usable.

(2) Approach Goals

Theme: Demonstrate the benefits of using a practical megaprogramming approach that combines the best of the STARS and RICC technologies to build the system

- Apply the RICC technology to SCAI development and demonstrate its benefit.

  The team will establish the benefits of using the RICC technologies in terms of reduced cost and schedule to develop and maintain the system. The amount of code automatically generated will be tracked, as well as the effort and schedule to obtain it. Cost associated with generated code will be compared with cost to develop hand crafted code. The team will monitor software problem reports produced for each code type, and track the reuse of RICC artifacts.

- Institute cooperating Domain Engineering and Application Engineering (DE/AE) processes including those for ongoing process improvements.

  There are many benefits attributed to a process-driven domain specific reuse approach to developing systems. These benefits include reduced cost, shortened development time, increased productivity and decreased risk. The SCAI Demonstration Project is an opportunity to invest in furthering a megaprogramming approach in the SWSC's command and control system domain. Given that there is an established C^2AI product-line, the intent of the AF/STARS SCAI Demonstration Project is to demonstrate the benefits associated with domain-specific reuse across the C^2AI domain, establish a process for continuously uncovering reuse artifacts and formalizing the domain models and architecture.

- Instantiate a process-driven SEE to support megaprogramming.

  Build a Software Engineering Environment that guides the engineering team in following their processes to create quality work products. The SEE must be affordable, and used by developers to do their work. It needs to adhere to standard open interfaces, and have a significant percentage of the commercially available products available on the IBM RISC 6000 and at least two other competing vendor platforms. New capabilities will be able to be easily added and users of the SEE will actively participate in identifying SEE improvements.

- Demonstrate that the SCAI application was built "cheaper, better, and faster" using the new process.

  The SCAI project is in a unique position to demonstrate that applications built using megaprogramming technologies are built "cheaper, better, and faster". The architecture megaprogramming work that had already been done, when the project started will offer a unique opportunity to measure the benefits of reusing the domain assets.

(3) Institutionalization Goals

Theme: Initiate the institutionalization to the SWSC megaprogramming paradigm used to develop the SCAI.

- Demonstrate widespread dissemination of technology expertise within the SWSC organization and its contractors.
- Establish a product-line organization and infrastructure, managing the evolution of multiple application systems.

  The organization will have a product-line manager responsible for approach, domain assets, process driven approach, and technology support and evolution. Users of the system will be an integral part of the total development/maintenance life-cycle.

  There will be a domain asset library and processes in place for managing and controlling the C^2AI and associated generators, reusable components, models, process assets, archives, and generic plans and cost models. The SCAI will gain commitment for an additional system to be managed by the product-line organization.
- Transition technology/assets to another organization for their own pilot application.

1.3 Organizational Structure

The SCAI project is under the auspices of the Plans and Engineering directorate (SMX) of the Space and Warning Systems Center (SWSC). The SWSC is a direct reporting unit under AFSPC Communications and Computer Systems Directorate. SCAI will use the SWSC expertise as required to help ensure the successful accomplishment of the AF/STARS project.

There are a number of government agencies as shown in Table 9 which will interface with the SMX to support the SCAI project. Table 10 lists the contractors directly performing the SCAI project.

Government Interface	Purpose
USSPACECOM Command Center personnel & AFSPC SC, DR & XP	Mission Expertise
Advanced Research Projects Agency (ARPA) with the STARS program office	Megaprogramming
Defense Information System Agency (DISA) Center for Information Management	Information Engineering
Software Engineering Institute (SEI)	Software Engineering Process
Embedded Computer Technology Resources Support Improvement Program (ESIP)	Command, Control, Communications and Intellegence

TABLE 2. Government Interfaces

Contractor Interfaces	Purpose
Loral Federal Systems	Technology and support
Kaman Science Corp.	Engineering analysis and Application Engineering
PRC	Information engineering and process definition
Rational	Ada programming environment, process definition and Booch methodology expertise
CACI	Domain Engineering and process definition
TRW	C^3I architectural infrastructure (RICC), process definition (TRW Ada process model) and systems integration

TABLE 3. Contractor Interfaces

1.4 Overview of Technical Approach

The Air Force's long-range objective is to realize a product-line organization, in which a wide range of its applications are developed and maintained using a coherent megaprogramming process. The partnership with STARS on the SCAI Demonstration Project is helping the Air Force accelerate its progress toward this objective.

The SWSC is responsible for the maintenance of a wide range of space and warning C^2 applications. The SWSC's long-range intent is to manage as many systems as possible within this domain of applications using a coherent process; and the envisioned product-line organization is intended to facilitate this type of management. There are three fundamental prerequisites for a successful product-line organization: a domain manager responsible for all applications in the product-line, sufficient commonality among the applications to allow management of the entire set as a whole, and enabling technology to facilitate the systematic management and engineering work.

To prepare for this type of organization, the SWSC is working with STARS to mature its technological approach, using the SCAI Demonstration Project as its primary vehicle.

Figure 2 depicts the SWSC's current concept of the future product-line organization and illustrates the role of the STARS megaprogramming technology areas in working towards this organizational objective. Appendix C has further discussion of this topic.

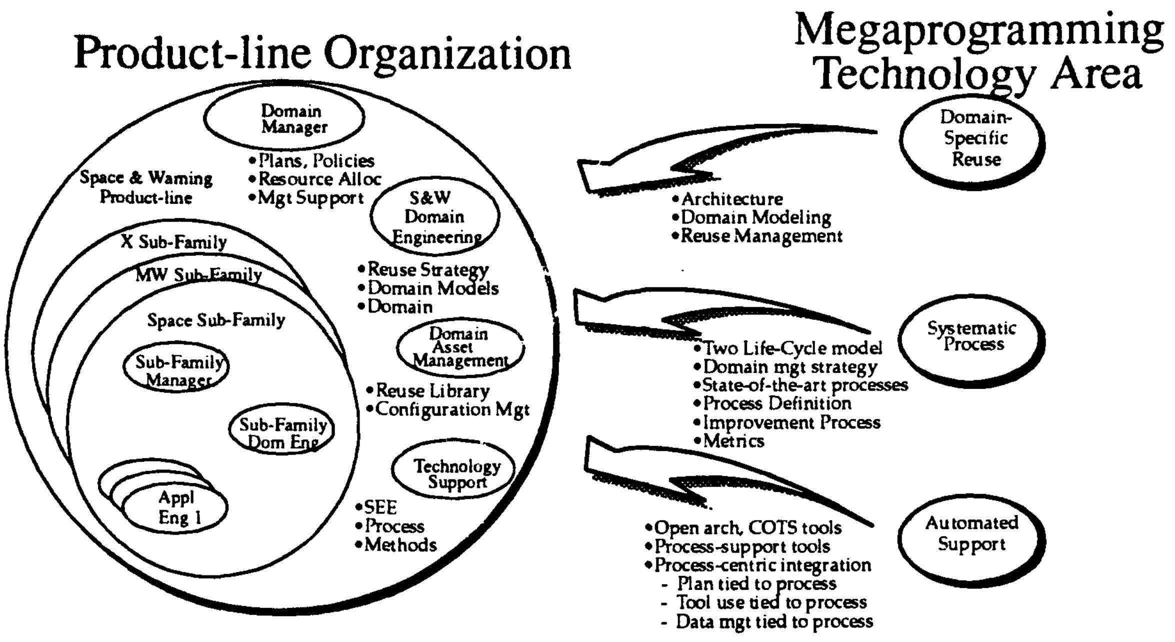


Figure 2. Megaprogramming: Enabling a Future Product-line Organization

This Experience Report includes four major sections providing the project's experience during the Preparation Phase of the Demonstration Project:

"Section 2.0, Domain Engineering/Application Engineering"

The central theme of STARS megaprogramming is enabling systematic reuse across a product-line of application. STARS advocates that the product-line organization institute a Two Life-Cycle process model, with Domain Engineering determining how to exploit the commonalities across the applications, and Application Engineering developing each application in accordance with the common approach and common assets identified by Domain Engineering. One of the key lessons learned by the SCAI team during the Preparation Phase is that Domain Engineering and Application Engineering are identical in many respects - so much so that it proves awkward to discuss them separately. For this experience report, they are treated as a single topic.

Prior to the formation of the partnership with STARS, the SWSC had already recognized the importance of domain-specific reuse and had launched a number of informal Domain Engineering activities. One of the key objectives of the partnership was to transform these informal activities into a systematic process. During the Preparation Phase, the team succeeded in formulating the SCAI DE/AE approach and defining an overall process architecture. This experience report topic deals with the work that was done during the Preparation Phase to define the approach. It also provides a brief description of the resulting approach itself. By the end of the Preparation Phase, the team had begun formal process definition work, based on the process architecture. The general-purpose techniques being used to define the DE/AE process are discussed in the Process Support section of the report.

"Section 3.0, Process Support"

The STARS concept of megaprogramming calls for concerted attention to process: in order to enable a product-line with effective domain-specific reuse, a systematic process (including a process-improvement process) is required.

Whereas the prior section discussed the work done in the Preparation Phase to define the Domain Engineering/Application Engineering process, this section deals with the technology used to support that process definition - as well as exploratory work that was done to prepare for process enactment.

"Section 4.0, SEE Support"

Megaprogramming, according to STARS, requires a significant level of SEE support to provide automated assistance to the product-line's process definition and enactment. During the Preparation Phase, the Loral/STARS team continued work on a process-support tool set that is intended for use during the Performance Phase. In the absence of full definition of the SCAI process, it was not possible to finalize the SEE configuration. Certain assumptions about the process were possible, however, allowing the team to install an initial increment of SEE workstations and to populate them with software tools to support key process activities. In addition, a preliminary SEE integration strategy was established, centered around AIX Workbench as the underlying tool-integration framework and the STARS-sponsored process-support tool suite as the basis for process control integration.

"Section 5.0, Metrics"

Metrics are instrumental in providing information to quantify software life-cycle cost, quality, and capability differences resulting from the presence and absence of megaprogramming technologies. The SCAI project will quantitatively and qualitatively measure the effects of megaprogramming on the development effort and the resulting product. During the Preparation Phase, the SCAI team formulated a preliminary metrics process. The foundation for the metrics process is the formally defined set of project goals. The team defined the project goals (refer to Section 1.2, Vision, Mission and Goals) and began detailing the metrics collection and analysis approach.

The remainder of the Overview section provides additional background information to set the context for the other sections of the experience report. Applicbility of Demonstration Project Goals to Key Technology Areas relates the Demonstration Project goals to the above defined experience areas. Iterative Technology Assimilation and Evolution: Applicability of the STARS CRFP discusses the inherently iterative nature of formulating a coherent technical approach - a principle espoused by the STARS Conceptual Framework for Reuse Processes (CFRP). Background Technologies and Tools then enumerates some of the specific technologies and tools that were assessed, adapted and combined in order to define the approach. Finally, Project Schedule presents an overview of the project schedule.

Applicability of Demonstration Project Goals to Key Technology Areas

Table 11 lists the Demonstration Project goals presented in Section 1.2 and indicates how each of the four major experience areas address them. In this table, the letter P is used to designate goals that are considered primary focuses of the area; S is used to designate goals that are supported to a significant degree by the area.

Demonstration Project Goal	Experience Area			
	Domain / Application	Process Support	SEE Support	Metrics
PRODUCT GOALS				
Develop a space mission capability suitable for operational deployment by October, 1995.	P		S	
Develop reusable domain assets	P	S	S	S
APPROACH GOALS				
Apply the RICC technology to SCAI development and demonstrate its benefit.	P	S	S	S
Institute cooperating Domain Engineering and Application Engineering (DE/AE) processes including those for ongoing process improvements.	P	P		
Instantiate a process-driven SEE to support megaprogramming		S	P	S
Demonstrate that the SCAI application was built "cheaper, better, and faster" using the new process		S	S	P
INSTITUTIONALIZATION GOALS				
Demonstrate widespread dissemination of technology expertise within the SWSC organization and its co 'tractors				S
Establish a product-line organization and infrastructure, managing the evolution of multiple application systems	S	S	S	S
Transition technology/assets to another organization for their own pilot application				S

TABLE 4. Applicability of Project Goals to Key Technology Areas

Iterative Technology Assimilation and Evolution: Applicability of the STARS CFRP

One of the key principles espoused by STARS is the Plan/Enact/Learn paradigm, as detailed in the STARS Conceptual Framework for Reuse Processes (CFRP), as illustrated in Figure 3.

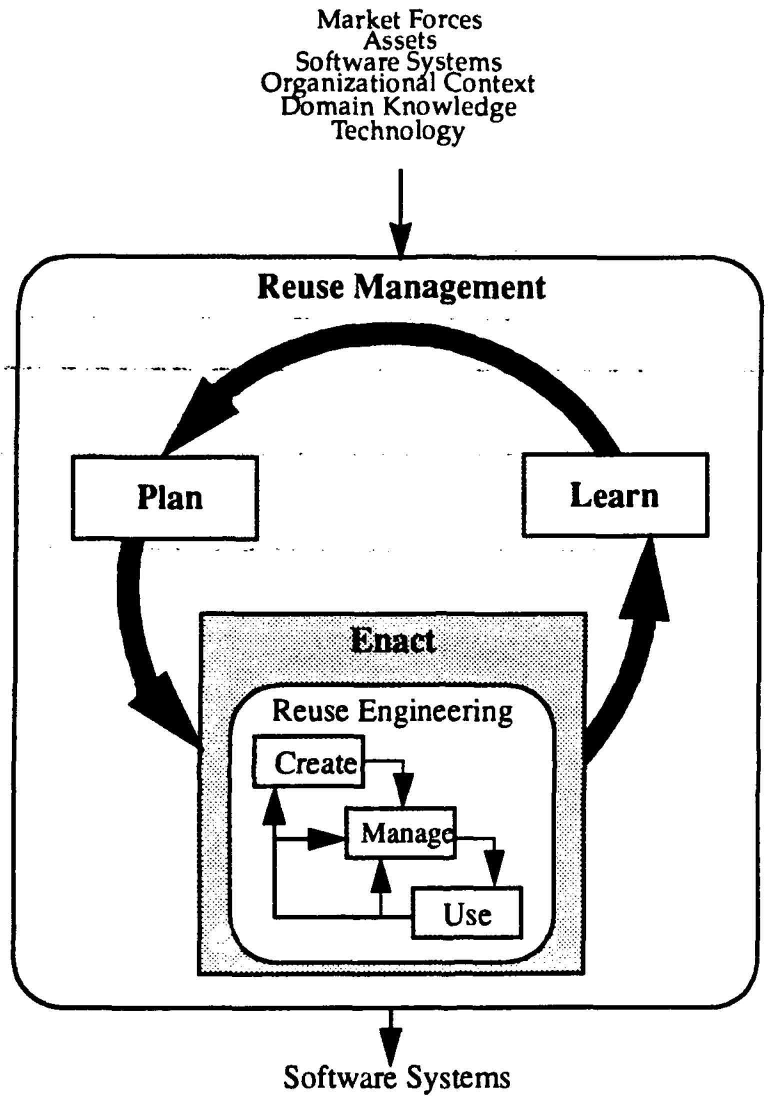


Figure 3. Conceptual Framework for Reuse Processes (CFRP)

The essence of this framework is that introducing anything significantly new is best viewed as an iterative undertaking; and that if the intent is to reuse the results, the iteration must be consciously managed. Thus, incremental changes are planned, the changes are applied, and the experience gained in applying the changes is used to adjust the plan for the next iteration.

For the SCAI project, it is meaningful to consider the CFRP from various points of view, notably:

(1) Formulating an overall technical approach,

(2) Formulating specific approaches for each of the three megaprogramming technology 'hrusts,

(3) Developing reusable assets for the domain, and

(4) Developing the SCAI application, including reuse of domain assets.

The first two deal with technology transition and process definition, which were the main activities addressed by the team during the Preparation Phase. The last two deal with producing application and domain products. At first glance, the CFRP seems to pertain primarily to items 3 and 4, but the SCAI team's experience has demonstrated that it in fact pertains equally well to items 1 and 2. If this fact had been better appreciated at the outset, the team might have been able to achieve consensus on the approach more quickly.

In general, the team has found that an iterative approach permeates nearly all significant technical activities.

Reviewing the experience across all major Preparation Phase activities, the SCAI team has concluded the following:

Lesson-Learned: The Plan/Enact/Learn paradigm, elaborated in the STARS Conceptual Framework for Reuse Processes (CFRP), applies to all major project activities - including the formulation of the project's approaches and processes as well as the development of its domain and application products.

Recommendation: Since an organization seeking to transition to megaprogramming can anticipate a significant amount of technology transition and integration, it should consciously plan an incremental build up of its approach.

Refer to Appendix D for a more detailed discussion of the Preparation Phase experience that has led the team to these conclusions.

Background Technologies and Tools

Formulating the SCAI technical approach involved evaluating a number of state-of-the-art technologies and tools, deciding which were most applicable, and determining how best to adapt and combine them. Although this proved to be a significant challenge, by the end of the Preparation Phase the team had agreed on the overall approach and had defined a preliminary process architecture.

Table 12 identifies some of the more significant technologies and tools that were considered. All of these technologies and tools are discussed in the four major experience report sections that follow. In addition, Appendix B provides further details for those items flagged with an asterisk (*) in the table.

The table divides the technologies and tools into three categories. Incumbent SWSC Technologies and Tools are those that were either already in use at the SWSC or had been pre-selected by Air Force management. STARS-Sponsored Technologies and Tools are those that had been developed or adopted by STARS as enabling technologies in support of megaprogramming. Jointly Developed Technologies and Tools are those that were formulated during the Preparation Phase by the AF/STARS team.

	Incumbent SWSC Technologies and Tools	**STARS-Sponsored Technologies and Tools**	**Jointly Developed Technologies and Tools**
DE/AE	CIM IDEF_0Modeling * Mission Operation/Information Analysis (MOIA)* Ada Process Model* Booch Object Oriented Analysis* RICC-based Architectural Infrastructure*	Conceptual Framework for Reuse Processes (CFRP) Two Life-Cycle Model for Coordinated DE/AE Processes Domain Analysis Process Model (DAPM)* Cleanroom Software Engineering Process *	
Process	Corporate Information Management Process Initiative (CIM) IDEF_0Modeling *	Process Driven Management and Engineering Approach * ETVX Process Modeling	Information Organizer Templates (with SEI)*
SEE	Reusable Integrated Command Center (RICC) Tool Set* Rational Ada Support Tool Set	Software Process Management System (SPMS)* CAT/Compass* Project Catalyst* Process Weaver*	
Metrics		Amadeus*	Goal-based Metrics Process

TABLE 5. Background Technologies and Tools

Project Schedule

The Demonstration Project is performed during the three phases described below and shown in Figure 4. The three phases are:

(1) A **Preparation Phase** where the goal is to ensure that all the technologies, assets, and training required to support use of the megaprogramming paradigm are in place.

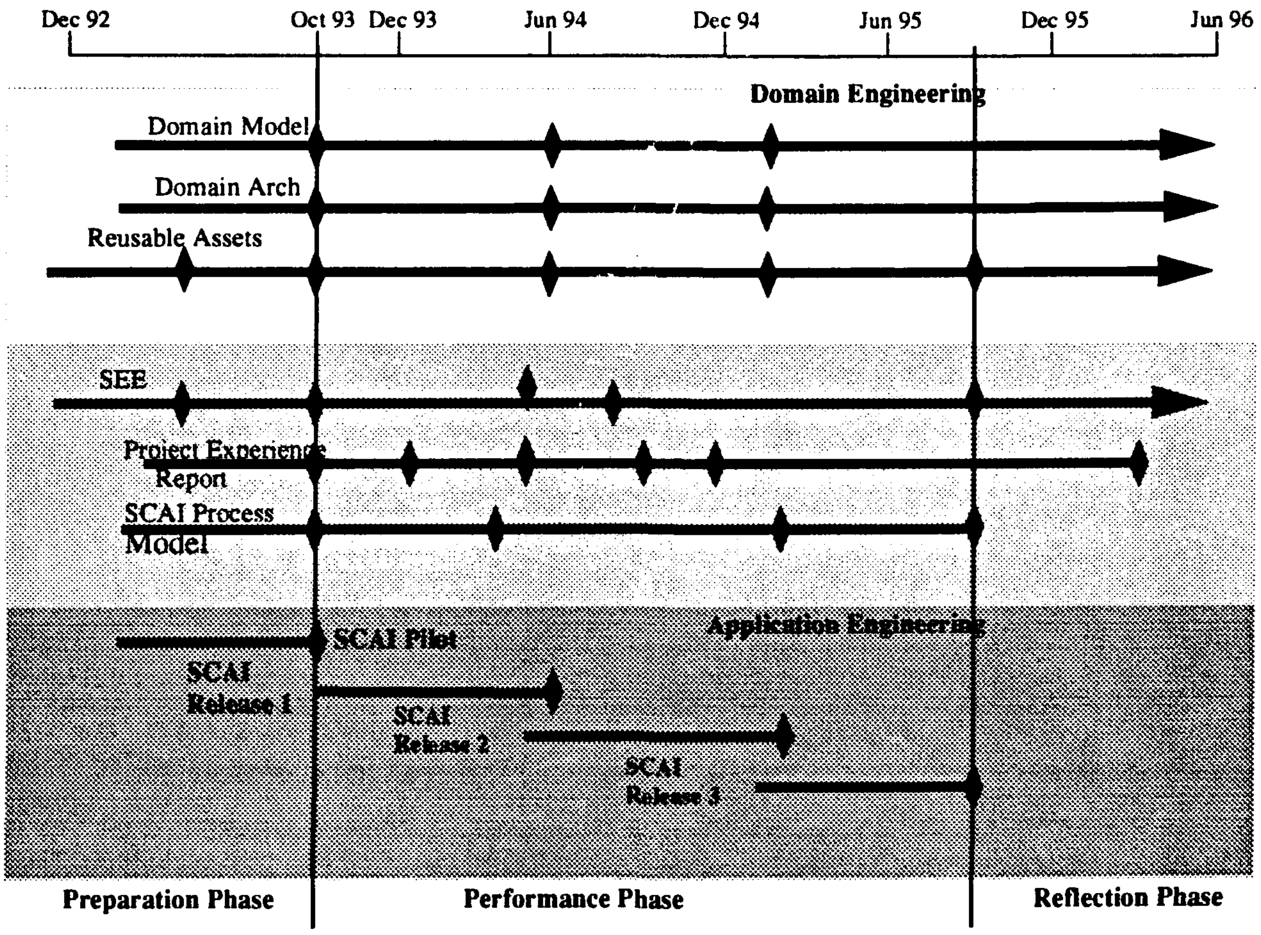


Figure 4. Project Schedule

(2) A **Performance Phase** where the goal is to apply and measure the use of megaprogramming technologies to establishing a product-line organization with the SWSC, and implementing an operational space control capability for the CMAH.

(3) A **Reflection Phase**, where the goal is to produce reports and presentations that document the significant results from the Demonstration Project, including lessons learned that will help others in adopting the megaprogramming approach. Reports will also be generated which quantify or project the benefits obtainable from the use of megaprogramming.

The remaining sections of the document describe the major technology areas of the SCAI project, in terms of objectives, plans, experience, lessons, and recommendations.

2.0 Domain Engineering/Application Engineering

2.1 Introduction

The central theme of STARS megaprogramming is enabling large-scale systematic reuse across a product-line of applications. STARS advocates that the product-line organization institute a Two Life-Cycle process model, with Domain Engineering determining how to exploit the commonalities across the applications, and Application Engineering developing each application in accordance with the common approach and common assets identified by Domain Engineering. One of the key lessons learned by the SCAI team during the Preparation Phase is that Domain Engineering and Application Engineering are tightly integrated processes - so much so that it proves awkward to discuss them separately. For this experience report, they are treated as a single topic.

Prior to the formation of the partnership with STARS, the SWSC had already recognized the importance of domain-specific reuse and had been pursuing a number of informal Domain Engineering activities. One of the key objectives of the partnership was to transform these informal activities into a systematic process.

During the Preparation Phase, the team succeeded in formulating the SCAI DE/AE approach and defining an overall process architecture. This experience report topic deals with the work that was done during the Preparation Phase to:

(1) Conduct precursor engineering activities, consisting of:

- Generating a pilot application, and
- Generating domain/application models; and

(2) Define the Domain/Application Engineering process.

The report also provides an outline of the resulting approach itself which embodies much of the teams overall experience.

The general-purpose techniques being used on the SCAI project to define the DE/AE process are discussed in a separate experience report section, entitled Process Support.

Long-Range Objectives

The following is a recap of the long-range project goals that are relevant to Domain/Application Engineering.

(1) Product Goals

- Develop a space mission capability suitable for operational deployment by October, 1995.
- Develop reusable domain assets.

(2) Approach Goals

- Apply the RICC technology to SCAI development and demonstrate its benefit.
- Institute cooperating Domain Engineering and Application Engineering (DE/AE) processes including those for ongoing process improvements.

(3) Institutionalization Goals

- Support the establishment of a product-line organization and infrastructure, responsible for managing the evolution of multiple application systems.

Relationship to Other Project Areas

Figure 5 depicts the relationship of the activities discussed in this section to other project areas.

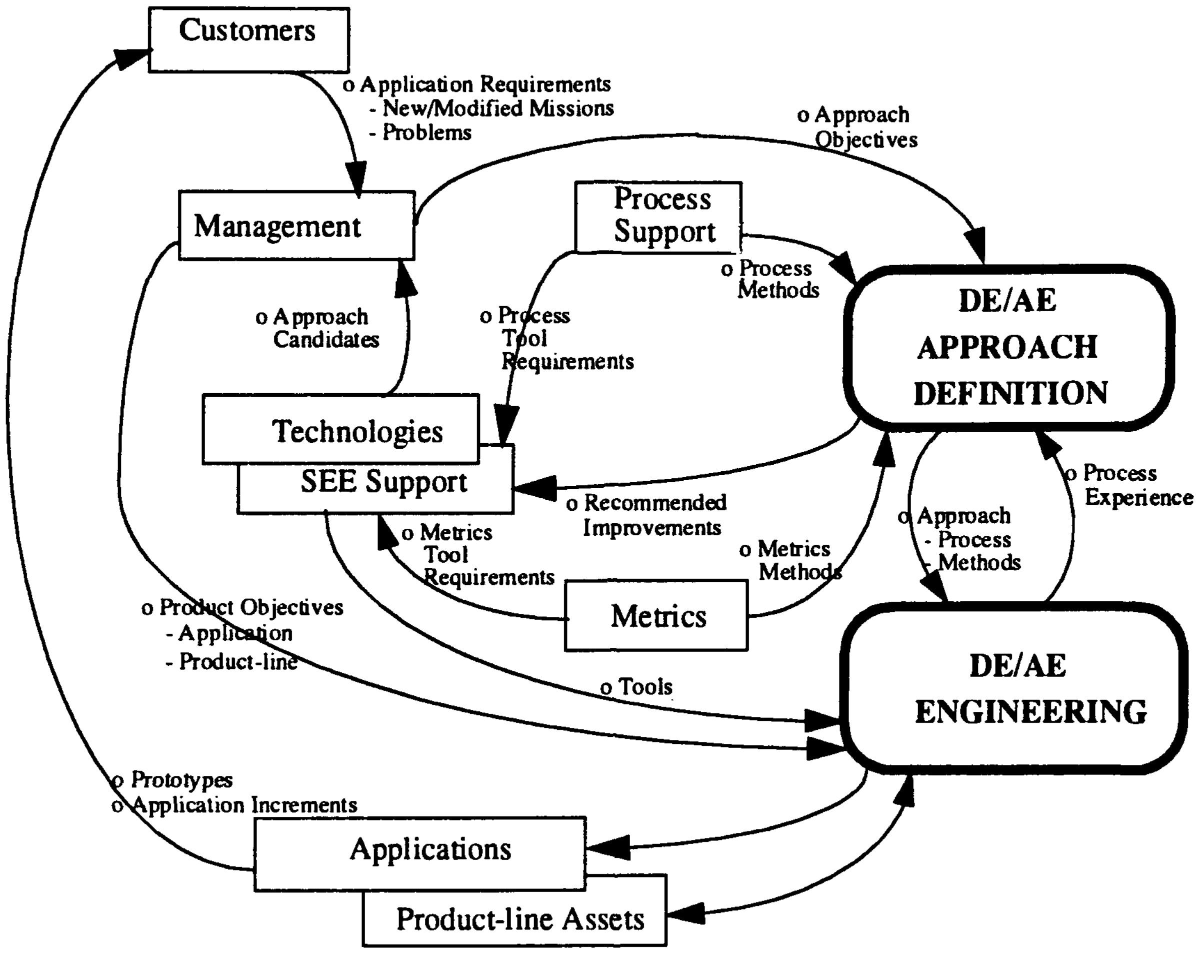


Figure 5. Relationship of DE/AE Activities to Other Project Areas

The highlighted boxes in the figure are covered in this section.

2.2 Preparation Phase Analysis

This subsection discusses the DE/AE experience gained during the Preparation Phase of the SCAI Demonstration Project.

2.2.1 Plans

Preparation Phase Objectives

Based on the long-term DE/AE goals discussed in Section 2.1 above, the Preparation Phase objectives identified for this area were:

(1) Near-Term Product Objectives

- Verify the RICC-based architectural infrastructure in the Space domain.

- Construct preliminary domain/application models.

(2) Near-Term Approach Objectives

- Formulate a DE/AE approach, including a preliminary process architecture.

(3) Near-Term Institutionalization Objectives

- Improve domain expertise.
- Learn the technologies and tools to be used in the DE/AE approach.

Assumptions

The following were some of the key assumptions made by the team at the outset:

(1) Use and improve the existing Architectural Infrastructure approach.

The 1990 Air Force Space Command (AFSPC) drive to develop a strategy to move to an architecture based on Open Systems Environments resulted in the development of a set of architectural components, called the Command and Control Architecture Infrastructure (C^2AI). Leveraging work accomplished by TRW on a production contract called CCPDS-R (Command Center Processing and Display System Replacement), the SWSC contracted TRW to advance CCPDS-R techniques and demonstrate a subset of the Cheyenne Mountain Missile Warning requirements, on a pilot program called the Reusable Integrated Command Center (RICC).

Significant commonality exists between C^2 systems, and the RICC architecture infrastructure developed reusable software to address these common requirements. The informal analysis of the C^2 domain resulted in a two layer architecture, with a service layer supported by a set of Ada Program Generators.

The foundation for Domain Specific Reuse for the STARS/SCAI project, consequently, is the Reusable Integrated Command Center (RICC) architectural infrastructure and program generator technology. See Appendix B for more information on the RICC approach.

A major assumption for the Demonstration Project was to take advantage of the RICC approach - including gathering additional data about its viability in this domain, and improving the capabilities and maintainability of the associated tool set.

(2) Use the modeling work already in progress at the starting point for the domain/application analysis.

Several key modeling activities were already underway prior to the start of the Demonstration Project. The Corporate Information Management (CIM) model deals with the general operational requirements for various Cheyenne Mountain command centers, while the Mission Operational/Information Analysis model (MOIA) focuses on the command center operations of the space support systems in particular.

These modeling activities were judged to be essential ingredients in any realization of the SWSC Domain Engineering approach, and the SWSC decided that they would be continued and that the work products would be incorporated as part of the domain modeling strategy.

Refer to Appendix B for a more detailed description of these modeling approaches.

(3) Use the Ada language for all software development.

(4) Continue reliance on the Rational Ada Support Environment.

Given the commitment to Ada, the SWSC had already established a close relationship with Rational and had decided to adopt the Rational Ada programming support environment and associated tools for the SCAI project. At first glance, this appears to be an assumption that is relevant only to the SEE, but it also proved to be influential to certain methodological decisions made during the Preparation Phase - such as the specific object oriented method to be used to develop the space domain model (the Rational Booch method, supported by the Rational/ROSE tool).

(5) Capitalize on certain STARS-sponsored concepts and technologies:

- Domain Analysis Process Model (DAPM),
- Conceptual Framework for Reuse Processes (CFRP),
- The Two Life-Cycle paradigm for coordinated Domain and Application Engineering processes, and
- The Cleanroom Software Engineering Process.

These concepts and technologies are discussed in the following subsections. Also refer to Appendix B for more details.

(6) Iterative nature of Software Engineering Process

The resulting approach also needed to allow for iterative development of systems, with ample opportunities for early buy-in to the design by the customer. The approach must also support product-line development, and be evolutionary in nature.

Planned Activities

Given these assumptions, the activities to be used to accomplish the Preparation Phase objectives were:

(1) Develop the SCAI pilot,

(2) Build initial models in the C^2 and Space C^2 domain:

- Space C^2 operations as currently practiced in Cheyenne Mountain,
- Capability model of the current SPADOC application software, and
- Information model of the current SPADOC application software; and

(3) Develop and document the DE/AE approach.

Table 6 depicts how these activities were intended to address the Preparation Phase objectives.

As shown in Figure 5, above, the DE/AE work can be thought of as two interacting activity groups: activities designed to formulate the approach (DE/AE Approach Definition), and activities designed to develop engineering products (DE/AE Engineering). Recognizing the need for usable products at the conclusion of this phase, a conscious decision was made to perform both activity groups in parallel. The early engineering work was intended both to provide experience needed to develop the approach and to produce work products that could launch the production phase work.

It should be noted that in view of the large amount of learning that faced the team, it was not possible to lay out a very detailed schedule. The plan boiled down to fairly loose statements of objectives in each area, with a deadline of 10/1/93 (the nominal start of the Performance Phase).

Some of the most profound objectives were institutionalization objectives: improve domain expertise, and learn the technologies and tools. Each of the planned activities would contribute to these organizational learning objectives.

Preparation Phase Objectives	Planned Activities: SCAI Pilot	Planned Activities: Initial Requirements	Planned Activities: DE/AE Approach
NEAR-TERM PRODUCT OBJECTIVES			
Verify the RICC-based architectural infrastructure in the Space domain	X		
Construct preliminary domain/application models		X	
NEAR-TERM APPROACH OBJECTIVES			
Formulate a DE/AE approach, including a preliminary process architecture			X
NEAR-TERM INSTITUTIONALIZATION OBJECTIVES			
Improve domain expertise	X	X	
Learn the technologies and tools to be used in the DE/AE approach	X	X	X

TABLE 6. Relationships between Preparation Phase Activities and Objectives

A key challenge facing the team was the fact that the SCAI project brought together multiple organizations, each with successes in using technologies with which they were familiar. This required that the project members had to understand the variety of SWSC incumbent technologies, the organizations in which those technologies were used, and the STARS technologies, before they could begin the process of integrating the technologies into the SCAI technical approach.

In view of the anticipated difficulty in achieving consensus on the DE/AE approach, the Air Force project management was faced with a dilemma. Working out the approach seemed to require the most talented, most experienced people. Yet if these people were devoted exclusively to the approach definition work, the product objectives could be in jeopardy. On the other hand, in the absence of a well-defined approach, there was a strong risk that the products developed during the Preparation Phase would not provide a sound basis for the Performance Phase work.

The management approach adopted by the Air Force was to keep its product-oriented activities relatively isolated from each other - and from the approach definition activities as well. Each product-oriented activity would be kept apprised of the maturing approach strategy but would be allowed to evolve its own detailed methodology. Similarly, the approach definition activity would be kept apprised of the results being achieved in the product-oriented activities and would attempt to take advantage of their experience in evolving the overall DE/AE approach. As confidence increased in the maturing DE/AE approach, more interaction between the various groups would be encouraged.

2.2.2 Summary of Accomplishments

The SCAI team successfully accomplished all of the major Preparation Phase DE/AE objectives cited in the prior section. The following paragraphs summarize the accomplishments.

(1) Developed the SCAI pilot

The pilot implements several significant functions present in the existing SPADOC application using the new open-systems based architecture infrastrusture. It has served as an excellent communication vehicle to the future users of the SCAI application; it has also provided a concrete demonstration of the viability of the architectural approach to outside observers. Building the pilot has helped transition the RICC- based system construction methodology to additional team members, and it has also helped improve the team's space domain expertise.

The most significant result of the pilot activity is the additional confidence the SWSC now has that the architectural infrastructure is viable for multiple applications within the product-line.

(2) Built initial requirements models:

- The CIM and MOIA models produced during the Preparation Phase capture the space operations to a sufficient level of detail to support the Performance Phase work. These models have served as a good basis for communication with the future users of the SCAI application. (These models are defined in Appendix B, and they are also discussed in more detail later in this subsection).

- Built a capability and information models of the current SPADOC application software, with emphasis on identifying the essential abstractions in the Space domain.

  Two models were developed: an object oriented model and an information model. The models will provide the foundation for the SCAI specification effort during the Performance Phase.

  The initial intent of these models was to capture the existing SPADOC software, to serve as the basis for re-engineering much of the SPADOC functionality for the SCAI. As this modeling activity progressed, however, another key objective emerged: the need to recast the current functionality in a form that would serve the interests of other applications in the domain. This led to the emphasis on identifying essential abstractions in the models, including the decision to map the SPADOC functionality to a new object oriented view.

(3) Developed and documented the DE/AE approach

The team achieved consensus on the overall approach, which was documented in Version 2.0 of the SCAI Demonstration Project Management Plan (SDPMP), dated 10/15/93. The approach is reflected in the SCAI process architecture, documented in $IDEF_0$ form, and attached as an appendix to the SDPMP.

The remainder of this subsection presents the experience gained while accomplishing the above.

2.2.3 Analysis

This material is based on more detailed experience information to be found in Appendix E, attached to this document.

SCAI Pilot Experience

The SCAI pilot was designed to test and demonstrate the feasibility of using the RICC as the architectural infrastructure to support the space mission. Two representative space mission applications were implemented.

(1) Satellite Ground Tracks

(2) Sensor-Satellite Look Angles

The target hardware was the IBM RS6000 series workstation. The intent of the pilot was to implement the applications with the look and feel of SPADOC 4B displays and to have the software produce the same numerical results as SPADOC. Allowances were made to accommodate differences between RS6000 and the SPADOC 4B Megatek with respect to keyboards and X Window capabilities, and the pilot was permitted to make minor improvements in display format and functionality.

Kaman Sciences was teamed with TRW. Kaman was responsible for implementing the space application software, and TRW was responsible for implementing the user interface displays, the data base, and the input and output message processing using the TRW developed RICC and associated tools. The application software was coded in Ada.

To determine the functionality to be implemented in the pilot, the team iterated on a set of "threads", or scenarios that were defined in terms of coherent user interface activities. Several reviews were held with current SPADOC users present in order to gain concurrence that the threads defined would be a representative set.

The goal of writing a small but representative subset of space applications software in Ada and of using the RICC architectural infrastructure was achieved. Using satellite element set and sensor location/limit data from SPADOC, the SCAI pilot produced ground tracks, ephemerides, and sensor-satellite look angles for five satellites representing a cross section of the satellite catalog. The results matched SPADOC 4B results within a reasonable degree of accuracy.

Modeling Experience

(1) Operations Modeling

Following the CIM methodology, this project was conducted in two workshops: a Baseline workshop which established the scope and began the analysis of the AS-IS Space Control Mission, and an Activity Based Costing (ABC) workshop. The Functional Economic Analysis (FEA) workshop which would normally follow the ABC workshop was postponed until after Missile Warning Mission and the mission of the Command Center could be completed to allow the FEA to be conducted on the aggregate processes. The workshops developed, refined and validated the Space Control Mission AS-IS activity model, identified improvement opportunities related to the processes performed in executing the mission, developed a Space Control Mission TO-BE activity model incorporating improvement opportunities. This effort was accomplished by a team of users and SCAI project team members and was viewed as top-down perspective of the operational activities.

The MOIA approach of analyzing command center operations through detailed reviews of concept of operations, checklists, and other command center artifacts was applied to three distinct space related operations centers in CMAFB, 1CACS, SSC and SPADOC. The effort was recorded using the IDEF notation and was viewed as bottom-up.

The CIM model and the MOIA models were then compared to ensure that the top-down and bottom-up approaches merged. These effort resulted in two major accomplishments:

- The development of an operational model of the space mission which could be used for both the applications development and further understanding of the Space and Warning domain.
- The opportunity to acquaint the user with the STARS Demo project

(2) Space Domain Requirements OO Model

The process used by Kaman Sciences to build a Space Domain Requirements Model was developed incrementally and is a composite of the Booch, Shlaer-Mellor, Concept Maps, and DAPM (refer to Appendix B, Technologies Contributing to SCAI). Using existing SPADOC 4 system documentation, phase one constructed an unabstracted object oriented model of the existing system. Phase two mapped the unabstracted model into two layers: a service layer and an application layer. Each layer was comprised of a pseudo-Booch Logical Model. This two-layer model was consistent with the RICC architecture. It also met the DAPM starting point requirements of postulating an initial architecture for the domain of interest. The third phase, with reuse as an objective, identified new services, when analyzing redundancies in the unabstracted Concept Model, that could be mapped from the application layer into the service layer. The Booch Logical Modeling, used during the second phase, was modified. Ada PDL (program design language) was added to describe the behavior of the objects identified in the model, and to have the models be consistent with Cleanroom practices.

The need to exclusively use the DAPM formal domain analysis technique to model the space domain was deemphasized because the project postulated that from a functional software perspective the operational system (SPADOC 4) software was equivalent to the Space Control Domain. This assumption is reasonable because SPADOC 4 is a replacement for existing Space Surveillance Center (SSC) systems, and is by definition a superset of the Mobile system. As a result, the focus was to create Application Engineering artifacts that were consistent with Domain Engineering principles and could be easily generalized into Domain Engineering artifacts. The process design team was to ensure that the Application Engineering process was reusable as a Domain Engineering process with a broadened scope of the analysis.

(3) Space Domain Requirements Information Model

The information modeling effort began with extracting database design information from the SPADOC 4 system engineering documentation. That information was analyzed and used to develop a third normal relational database schema. A data dictionary was established and populated. The effort was developed using Enity Relationship Diagram (ERD) notation. Due to the vast amount of data to be organized the initial conceptual data schema has focused on defining all entities, relationships, keys and attributes for release one.

The Space Control Mission workshop also developed a data model using $IDEF_{1x}$ notation. This high-level model was developed based on the information flow depicted in the activity model and independently to the ERD.

The ERD model was compared against both the entities defined in the OO model and the CIM $IDEF_{1x}$ data model to ensure consistency. Although each of the models depict varying levels of detail and were developed from different perspectives and purposes they were found to be reasonably congruent.

This resulted in a conceptual schema which could be developed into database tables to allow the development of the application.

Approach Definition Experience

A great deal of experience was gained as a result of the activities in this key grouping. Because of the volume of experience data generated, only a summary is presented here. Appendix E, Preliminary DE/AE Process, is attached to this report to provide additional information for interested readers. The appendix covers the following topics:

(1) Selected DE/AE Approach Topics

(2) Creating a DE/AE Process

(3) Preliminary SCAI DE/AE Process

The following is a summary of the DE/AE Approach Definition experience gained during the Preparation Phase.

(1) Iterative Buildup of Approach

Forging a megaprogramming DE/AE process was perhaps the most difficult technical challenge faced by the SCAI team. This was due to the amount of learning required by each team member, and the difficulty in integrating the best of the strong candidate technologies, methods, and tools being considered. It proved impossible to define the process in one massive intellectual leap. The only way for the team to cope with the problem was to iterate. In retrospect, there was no formal process followed while this work proceeded. Working groups would alternate between periods of intensive interaction and periods of relative inactivity - while other more pressing issues were addressed. Frequently, several approach issues were being studied in parallel by separate groups. Discussions were sometimes contentious. In general, with so much to integrate, a great deal of "groping" occurred.

Despite the seeming confusion, however, some very bright and experienced people managed to make a great deal of headway, and at the conclusion of the Preparation Phase, the team had achieved a high degree of consensus on the overall approach.

Although there was no formal plan or process followed while this iteration was in progress, some patterns did emerge. In general, the implicit process was something like the following:

- Postulate a facet of the approach and attempt to elaborate it;
- Identify the most pressing issues associated with the postulated facet;
- Attempt to resolve the issues;
- If rough agreement can be achieved, attempt to articulate the agreement and document it or present it to a wider audience to gain consensus;
- If a sufficient level of consensus is achieved, attempt to merge the new facet into existing materials used to capture the approach (papers, charts, etc.).

By the end of the Preparation Phase, the approach was far from complete, yet sufficient agreement had been reached about the key aspects of the approach, that it was possible to launch the Performance Phase work.

In summary, the team recognizes that iterating on the approach is a fact of life, and further, that the iteration will continue throughout the life of the product-line (i.e., process improvement is a continuing responsibility.).

(2) Coming to Grips with Terminology Problems

One of the key problems impeding convergence on approach was terminology. Terms such as "architecture", "domain", and "model", had different meanings to virtually every person. A frequent scenario was a protracted, sometimes emotional interchange that culminated in a realization that the parties had been talking about two different things (or more!). A glossary would have been very useful, and in fact one gradually emerged. The difficulty in generating a glossary was that some terms continued to defy consensus. A practicle glossary appears in Appendix A. The move towards a complete glossary is continuing into the Performance Phase.

(3) Shlaer-Mellor Pilot Model

In an attempt to distinguish exactly what Domain Engineering might be for the SCAI project and to determine exactly what a Domain Requirements Model and a Domain Architecture Model would look like a four week effort was spent building an experimental Domain Requirements Model using Domain Analysis Process Model (DAPM) and the modeling views advocated by Shlaer-Mellor Object Oriented Analysis. The subdomain of Space Operations was selected for the modeling pilot, since the first release of SCAI software was intended for that domain.

The Shlaer-Mellor methodology advocated a layered model framework, which was also being advocated by the space analysis work being done. This notion of layering ended up being an essential aspect of the current approach (refer to Appendix E).

(4) Resolving Dissonant Aspects of Approach

With the number of technologies, methods and tools being considered, it was inevitable that aspects that were considered "mandatory" appeared to be unalterably dissonant.

As an example, the team had an objective of taking advantage of three key technologies: Ada Process Model, OO Analysis and Design, and Cleanroom Software Engineering (refer to Appendix B Technologies Contributing to SCAI, for an explanation of these terms). Each had undeniable advantages. Ada Process Model stressed building the system incrementally, with each increment disclosing more of the design, and each increment giving more insight into the true requirements. This seemed to be at odds with the Cleanroom principle of getting the requirements correct from the start, before proceeding to implementation. Cleanroom seemed to dictate functional decomposition, in opposition to the object oriented approach. And so on.

In fact, the team was able to forge an entirely new approach from the three - potentially strengthening each of the ingredient approaches. Although this integrated approach is still being elaborated, and although a lot more experience needs to be gathered, the marriage appears off to a good start. Refer to Appendix E, Preliminary DE/AE Process, for a high-level summary of the current approach, showing how the three technologies are used in concert.

Some of the other aspects of the approach that proved to be hurdles were:

- Mapping the STARS "Two Life-Cycle" process model to DE and AE
- Early thinking called for two entirely distinct processes: Domain Engineering, and Application Engineering - each with its own set of artifacts that were somehow tied together. At this point in time, the team believes that the processes are integrated as one process (hence the name of the current Experience Report section), and that many of the modeling artifacts are joined together.
- Requirements Models vs. Architecture Models
- Early thinking called for four distinct models: Domain Requirements Model (DRM), Domain Architecture Model (DAM), Application Requirements Model (ARM), and Application Architecture Model (AAM). The above paragraph indicates that the distinction between the Domain and Application models is blurred, and it now also appears that the distinction between the requirements and architecture models is also blurred.

(5) Capturing the Approach

Once a decision had been made about the approach, the question of how to record it emerged. In fact, much of the approach emerged on slips of paper, on whiteboards, on presentation charts, and (sometimes) in more formal white papers. All too often it occurred that two parties left a working session with a secure feeling that agreement had been reached, only to discover later that there were still differences. Writing the agreement down helped prevent this, but even that wasn't an ironclad solution.

Two main project-wide artifacts have been used effectively to help cement the team's approach. The first was the SCAI Demonstration Project Management Plan (SDPMP). Version 2.0 of this document, dated 10/15/94, brought together the overall approach in a fairly coherent fashion. The second was the SCAI Process Architecture, an IDEF$_0$ model of the top three-or-four levels of the SCAI process. This described the activities in the process as well as the primary interfaces among them; it also included a glossary of terms used in the process model.

As the team moves into the Performance Phase, the approach is being further refined into detailed process definitions. This refinement is being done incrementally, with each increment dictated by the timing of the Performance Phase activities and by the maturity of the area of the process. The first process area scheduled for detailed process definition is the specification process - a process that is heavily dependent on the modeling artifacts.

2.2.4 Lessons-Learned

The DE/AE lessons are divided into the following subcategories:

(1) Domain Engineering/Application Engineering Process Definition

(2) Creating a Space Domain Requirements Model

Given the current status of the project (transition between preparation and Performance Phases), these lessons should be viewed as partially validated. More definitive lessons will be provided in future versions of this Experience Report.

Domain Engineering Process Definition

The following lessons were learned while trying to create a Domain Engineering process that embodied STARS concepts as well as existing SWSC technologies.

Lesson: Interpret the STARS "Two Life-Cycle Model" as a single integrated DE/AE process.

The SCAI team has found it necessary to re-interpret the STARS "Two Life-Cycle Model" to show very close interaction between Domain Engineering and Application Engineering, in order to constantly validate the domain models against real applications in the domain. The models must be developed iteratively to avoid the creation of complete but unvalidated models. In fact, the team has developed a working hypothesis that application level models to a large extent should be views of domain level models.

Lesson: Domain analysis principles should be applied even to individual applications.

Domain Analysis typically looks at multiple systems with the intent of discovering the similarities and the differences. The overlap of DA with Systems Analysis should be in identifying system similarities, which is synonymous with identifying high level abstractions. If the DA process is well-understood it should be applicable to a single system. If DA is applied to a non-perfect, single system, the result should be a simpler, well-structured, single system. Thus, even if an organization is initially unwilling to invest in DA and generalized component development across the whole extent of the domain, they may be willing to apply domain analysis to a single system, and still be able to judge the economic and technical value of the DA process. This does not change the purpose of domain analysis from identifying commonalities across multiple systems.

Lesson: Domain analysis must include operational analysis of system usage.

Domain Analysis typically examines similar software systems by examining the requirements, design, and software of these systems. However, to create truly abstract domain requirements (problem-state model), the scenarios via which users interact with the systems must be understood. The entire process that the organization follows developing software systems may also need to be examined in order to create an abstract Domain Requirements Model. For example, in the Space Domain, several systems involve the Orbit Analyst in a complicated dialogue across multiple display screens in order to determine the orbit of a satellite that requires non-routine calculations. Two reasons for creating the dialogue are: 1) The calculations are too expensive for obsolete hardware to handle on a routine basis, 2) Compute-expensive routines would have to be written to duplicate the intellect of Orbit Analysts inspecting on-screen graphics. As hardware becomes faster, the need to involve the Analyst in an external dialogue may disappear. However, without examining the Orbit Analyst's role in the overall organizational process, the reason for the specific Human-Machine Interfaces, which are common across all the Space Systems, would not be clear.

Lesson: The specific DA process may not be as important as planning and assigning domain experts and systems engineers to search for system commonalities.

Without a formal process, a set of common services for C^2AI systems was developed by TRW. These services are embodied in the RICC products. Independently, Ruben Prieto-Diaz applied his DA process to the same domain at Contel. Independently, IBM developed its CCS-2000 Domain- Specific architecture. In all three cases, the *same* common services were discovered. The lesson that might be learned is: The specific DA process used to uncover the common services is not as important as the fact that there is a planned and concerted effort to discover system commonalities.

Lesson: Class models can be used as domain models.

Booch Class Models are good mechanisms for recording Domain Requirements Models. Superclasses, class categories, and polymorphism intrinsically identify high level abstractions/system commonalities. Subclasses identify system differences.

Lesson: Domain analysis should produce the simplest reusable system.

Domain analysis should not just statically identify reusable artifacts, but should identify reusable artifacts that will result in the simplest possible reusable system in the domain. One method for identifying the simplest possible system is to compare the behavior encapsulated within classes to the interfaces defined between classes, as represented by object scenario diagrams. If identifying a common reusable class results in a scenario of usage that sends a single message into and out of a common reusable class for all systems in the domain, and the reusable class abstracts complicated internal behavior, then the reusable class is loosely coupled. If, however, the scenario of usage for the reusable for class across all systems in the domain results in hundreds of messages in and out of the class then the class is tightly coupled to other classes and is not a good choice as a potential reusable component abstraction.

Lesson: Domain Requirements Model for C^2AI must be layered.

Two philosophies of domain-specific reuse existed within the SEI: the D'Appolito school, which believed in basing reuse on a common Domain Architectural Model, and the Cohen school which believed in basing reuse upon both a Domain Requirements Model *and* a Domain Architectural Model. Through practice, we accepted a Cohen Approach modified by Prieto-Diaz. A requirements model shows what problem systems in the domain are supposed to solve. When technological changes mandate a different architecture, systems need not be redeveloped from scratch. Big DoD systems are going through a technology shift right now with networked workstations replacing large mainframe computers. The next such technology transition might be the use of massively parallel computers. Though it is necessary to capture the domain problem in a mode using the Prieto-Diaz process of domain analysis, domain "artifacts" will be uncovered which do *not* relate to the problem state. As an example, references to querying a relational data base are *not* problem state artifacts, but *are* encountered frequently during analysis of C^2AI systems. The Prieto-Diaz approach is to postulate an architecture and provide a layer in which these references can be captured. The postulated architecture for the SCAI is the RICC "Chip" layered architecture. This layer need not be elaborated as part of the formal requirements model, but it can be developed and elaborated as part of the Domain Architectural Model. This service layer to the Domain Architectural Model will need to be elaborated to identify whether sufficient services exist to develop the application layer in the domain architecture.

Lesson: Domain software components should include validation data.

A Domain Architectural Model defines the context via which reusable components communicate. The STARS SCAI team feels that reusable specifications, validation, testing, and intended usage context are necessary for reusable software components. This contextual data is contained in Cleanroom Six-Volume Specifications, which are generated by the Cleanroom Process. Cleanroom System (Domain) Specifications will define a formal and implementation free, functional set of requirements. These requirements will define the boundary of the system. They will also provide a way of validating the Domain Requirements Model. If the same stimulus history applied to both the Domain Black Box and the Domain Requirements Model produce the same response, then the Domain Requirements Model has been validated.

Creating a Space Domain Requirements Model

In the following discussion, describing initial experiences in applying OOA modeling principles, some non-standard terminology is used. One of the techniques used is called "Concept Maps", which is an informal brainstorming technique in which the modeler is not restricted to fixed object relationships, eliminating redundant entities, or making clear distinctions between entities and attributes. OOD refers to a Booch logical analysis process, with two caveats: 1) Object Scenario Diagrams do not show whether message passing is synchronous or asynchronous, 2) Object Scenario Diagrams have Ada PDL associated with them to show high-level behavioral abstraction.

Lesson: Domain analysis must concentrate on capturing the "apparent" aspects of the domain and characterizing its essence, rather than its atomic details.

Although time constraints are always a factor in projects, the analysis of the entire space domain in the given time-frame was somewhat overwhelming. The time-factor forced us to deal with the "essence" of the domain and, rather than getting bogged down in the seemingly infinite details within the space mission, and we succeeded in capturing the entire domain and its essential behavior and relationships. Consequently, we were able to produce a domain-wide OOD that is merely missing a consistent depth of refinement. The level of depth within the analysis and design products is inconsistent. That is, some entities in the analysis model are detailed well and others are not. In the design model, we could not refine all of the class operations down to the same level, nor could we deal properly with concurrence, persistence, or synchronous/asynchronous messages.

Lesson: Behavioral Abstraction is necessary.

A method of depicting and sequencing behavior, other than through PDL is needed. Development of the architecture is driven by sequencing implications and if an object scenario has been refined to PDL there is inertia making design changes difficult. Performance design should take place earlier.

Lesson: Project terminology needs to be shared and defined in a Project Glossary that is well distributed throughout the project.

Terminology was a stumbling block For example, when we initially defined the process and the products we were going to produce, we used the terms "model" and "architecture" and carried these for a long time in the project. However, they later proved to be confusing to other SCAI members and thus counterproductive. Although we changed the terms to OO Analysis Model and OO Design Model late in the game (expensive in man-hours) to try to overcome the difficulties, even these terms are not without castigation, and a consensus on terminology is still elusive.

Lesson: The flexibility to change process definition real time is important.

The definition of a process was one of the absolute necessities for this modeling effort. It defined the ordered steps and activities with their accompanying inputs and products. We made several modifications to the process as we were going through it and that in itself is important. Having the flexibility to change the process for real-time improvement is important. It was essential not to treat the defined IDEF process activities as inflexible due to the amount of interaction that existed between activities and the detailed artifact definitions required.

Lesson: Both functional and object oriented mind sets are required.

Due to the functional nature of the SPADOC requirements, and the need to depict the control state aspects of the SCAI threads it proved necessary to use our functional mindset at times. However the learning curve for OOA/OOD is steep and thinking in terms of objects was not easily caught by those who have been using functional decomposition for many years. We were fortunate to have someone on our team who had several years experience in OOA/OOD. This proved to be an extremely valuable asset.

Lesson: Duplication of entities does not hinder understanding of focal entity behavior and relationships.

In developing the analysis model, we initially were confronted with the problem of duplication of entities. That is, when we decomposed an entity down one leg, we inevitably uncovered a relationship to an entity that showed up in another leg. The problem was this: either we accepted duplication and documented the same entity every time we encountered it, or we had to solve the larger problem of how to characterize focal entity relationships without duplication. We made the decision not to worry about duplication and proceeded to give each entity a unique identifier based upon its context in the mapping. This proved to be a good decision in that it reduced a great deal of coordination overhead and preserved the integrity of each concept map. Our objective was to capture the focal entity behavior and relationships and this allowed us to concentrate on that objective. It turned out that the duplication did not pose a problem in transitionally to the OOD phase.

Lesson: A single or chief architect who is responsible for assuring the integrity of the overall system architecture is essential.

When we entered the OOD phase, it became very clear that we needed to have an overall architect who would design the initial class diagrams and review all of the class and object diagram development. Without this, the project would have not succeeded. It was absolutely essential to maintaining a logical and consistent design. We also discovered that every team member had to have a total domain understanding in the OOD phase. This was a change from the OOA phase, where each analyst could take a particular leg of the analysis model and characterize the individual entities. However, in the OOD phase, the relationships between the classes and objects were much more pervasive and required a broader understanding of the domain.

Lesson: *The design process of stepwise refinement is valuable.*

Although not unique to OOA/OOD analysis, we found that following the design process of stepwise refinement very valuable. We developed the initial class diagrams with their operations and the initial object diagrams and then began the refinement process. As we continued this refinement, we discovered new operations, classes, relationships, etc., and simply updated the model accordingly. This process can simply be continued until you arrive at the code level following very easy and small incremental steps.

Lesson: *State transition analysis of operator sessions was valuable.*

One of the unique things that emerged from this effort was the characterization of the sessions through state transition diagrams. It would be hard to overstate their value to this effort. They provided a solution to capturing what appeared to be very complex display and operation relationships. We used these session display state transition diagrams as one of the basic foundations for the OOD phase. Sessions are now viewed as comprising a new layer to our Domain Requirements Model called the "Event" layer. This layer is very closely related to an operational concept for the SCAI system.

2.2.5 Recommendations

Integrated DE/AE Process: Interpret the STARS "product-line Model" to call for a single integrated DE/AE process.

Iterative Approach: Plan on a managed approach to iterative build ups of both process and product, including precursor piloting activities.

Layered Modeling Framework: Develop a modeling framework that separates layers of abstractions and follow this framework through both requirements and architecture modeling.

Project Glossary: Put early priority on establishing and using a process to maintain a glossary, to combat terminology disconnects in discussions about approach and process.

3.0 Process Support

3.1 Introduction

Prior to the STARS involvement, the SWSC had made significant strides in moving to an architecture based on open systems environment and developing practical approaches to generating reusable architectures and artifacts. They had made little attempt to define in detail the processes they used. As a result of using the Software Engineering Institute (SEI) Capability Maturity Model (CMM) to assess the SWSC, the organization recognized the need to develop a detailed definition of their processes. The Deputy Assistant Secretary of Defense for Information Management had also chosen to sponsor a Corporate Information Management Workshop to define the processes performed in conducting software maintenance at the SWSC.

The SCAI project has created a Process Support Team which is responsible for process planning, process definition, automated process enactment, process tracking, process measurement and process improvement. This team supports Project Management, Domain Engineering, Application Engineering and other project Support Teams, but each team is responsible for and retains ownership of their process. Process definition formalism is being strived for at the level where:

(1) Software process guidance is tied into tool invocation mechanisms and in logon and logoff scripts. Consulting on the current software process status is available.

(2) Process management user dialogues regarding planned and actual activities are provided, plus non obtrusive data collection, management reporting, and expert consultation.

(3) Proactive process management of process artifacts and activities based on knowledge of user roles and the software process is provided.

Long Term Objectives

The SCAI project outlined its long term objectives in the area of process as follows:

"Institute cooperating Domain Engineering (DE) and Application Engineering (AE) processes including those for ongoing improvement and instantiate a process driven Software Engineering Environment to support the SWSC TO-BE process."

Specific objectives include:

(1) Demonstrate process driven project planning.

(2) Improve a process as a result of being process driven:

- practitioners follow the process.
- process engineers are able to collect quantitative and qualitative metrics.
- metric data is analyzed.
- process improvement is planned and implemented.

(3) Evolve SCAI project into a process driven project development organization.

Following the SCAI project, the intent is to institutionalize this approach to doing business thought the SWSC.

3.2 Preparation Phase Analysis

3.2.1 Plans

Preparation Phase Objectives

During the Preparation Phase the SCAI project process objectives were to:

(1) Model and document the SWSC "AS-IS" process and then define the SWSC "TO-BE" process using IDEF$_0$ models and associated artifact definitions.

(2) Prepare the process support tooling for use in project planning and automation.

(3) Establish a method for formally defining and refining SCAI processes during the development phase of the project.

Planned Activities

The SCAI project contained several key processes which could not easily be represented as one process. The processes were:

(1) The SCAI Application Engineering Process - the collective process to be defined to support developers to produce the SCAI-based space command and control system.

(2) The SWSC Domain Engineering Process- The process to be defined to support the development of:

- models characterizing the space domain.
- an architecture that satisfies a class of surveillance data processing applications that support both the domain subfamilies of space, air and missile warning.
- reusable software components or applications to prepare pluggable software components.

(3) The SCAI Services Process - The process to be defined to provide support services, such as configuration management and software quality assurance for other SCAI processes.

(4) The SCAI Management Process - the process to be defined to support managers in the planning, delegation, dispatching, monitoring and controlling of processes and tasks on the SCAI project.

(5) The SWSC Process Improvement Process - the process to be defined to support the specification, design, development and improvement of SCAI processes.

Each of these processes satisfies a specific need. Each of these processes have defined interfaces and both request and receive information and services from other processes and as such can be viewed as peer processes. See Figure 6.

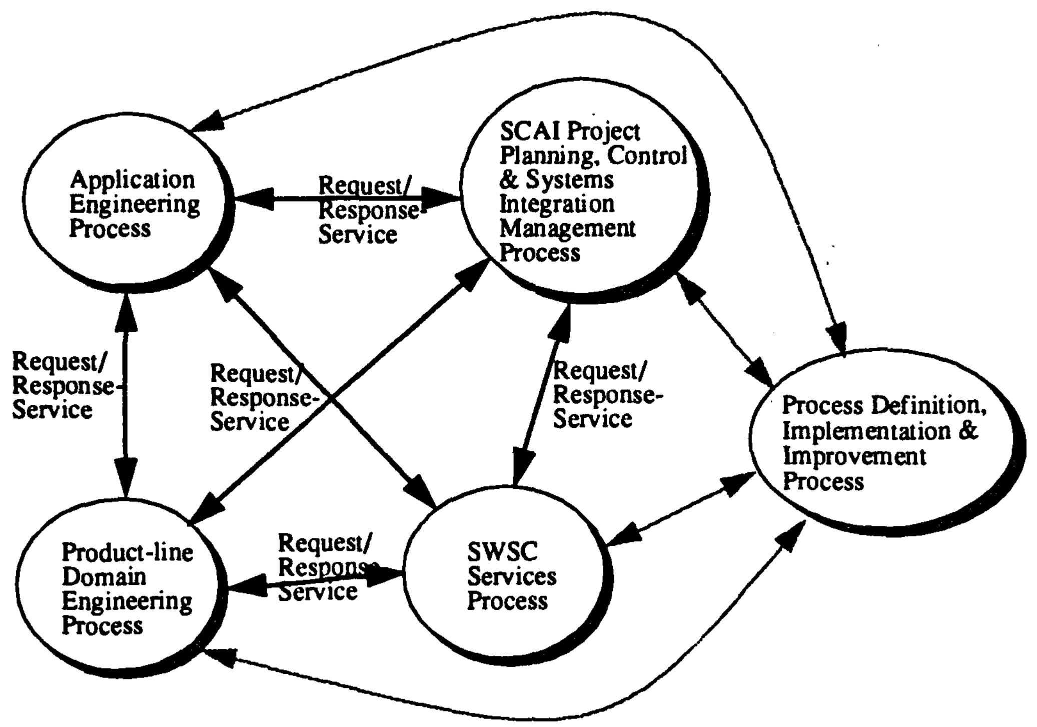

Application Engineering Process
SCAI Project Planning, Control & Systems Integration Management Process
Process Definition, Implementation & Improvement Process
Product-line Domain Engineering Process
SWSC Services Process
Request/ Response-Service
Request/ Response-Service
Request/ Response-Service
Request/ Response-Service
Request/ Response-Service

Figure 6. Process Architecture Concept

Each of the above processes are themselves composed of process components and can be represented as an architecture where the interfaces of each process component is defined along with the service set that each process component performs.

This organizational concept supported our specification of the SCAI Application Engineering process and was used to identify the key interfaces between the SCAI Application Engineering process and the other SCAI processes.

Within the SCAI Application Engineering process there are a number of key processes that were identified. To support the planning for SCAI Process Definition, the steps identified in Figure 7 were performed. A strawman workflow of the SCAI project activities was defined from surveying the relevant existing processes, and identifying how the SCAI project was to meet its defined goals and its unique project requirements. This workflow analysis was prepared using the IDEF_0activity modeling technique for representing abstractions of key project activities, while an ETVX view of the workflow provided basic control information for the process. SCAI contractors received briefings on both IDEF_0activity model and the ETVX workflow model and were involved in process reviews. The information collected from both views will be coalesced to form a SCAI AE process architecture from which an SPMS [1] based model of the SCAI process will be developed to support strawman SCAI project planning.

1. SPMS or Software Process Management System is the name used to identify the STARS-supported tool to support process specification and process-driven project planning and project monitoring. Its successor is the commercially-supported Process Engineering & Analysis Kernel System.

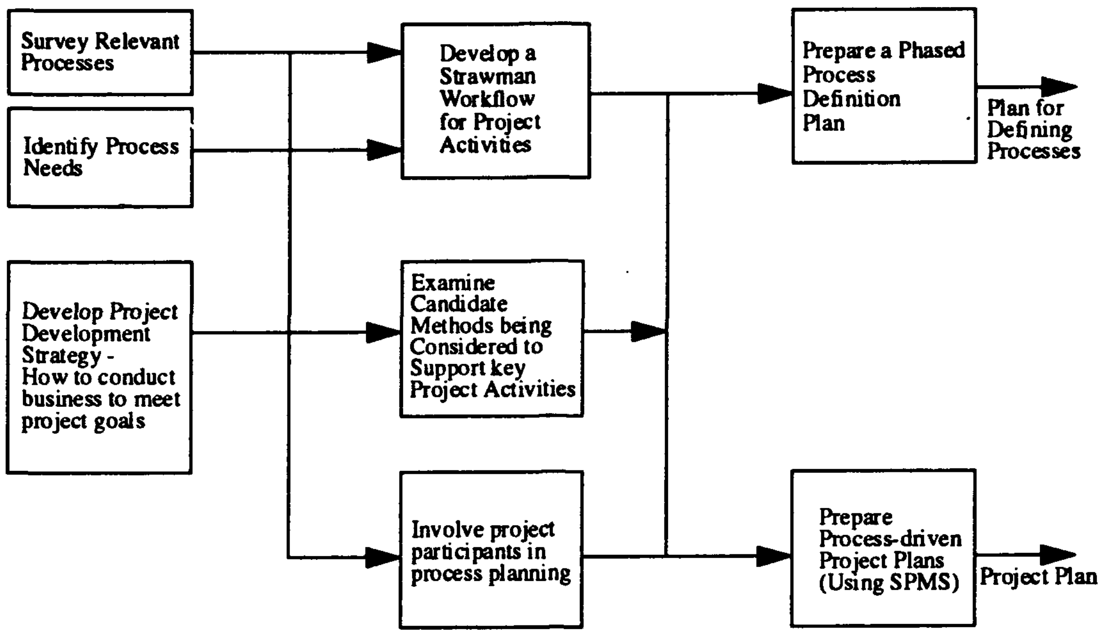

Survey Relevant Processes
Identify Process Needs
Develop Project Development Strategy - How to conduct business to meet project goals
Develop a Strawman Workflow for Project Activities
Examine Candidate Methods being Considered to Support key Project Activities
Involve project participants in process planning
Prepare a Phased Process Definition Plan
Plan for Defining Processes
Prepare Process-driven Project Plans (Using SPMS)
Project Plan

Figure 7. SCAI Process Planning Steps

The AE SCAI process architecture was used to prepare a phased process definition plan which was employed to determine the order in which SCAI processes would be defined using the Information Organizer Template[1] techniques for defining manually enactable processes. By performing the planning activities shown in Figure 7, the SCAI project will position itself to place key SCAI activities on a process-driven basis and provide the groundwork to actually plan portions of the SCAI project using the STARS concept of process-driven project planning.

Approach to Process-Driven Development

The SCAI approach to process-driven development began by first performing $IDEF_0$ activity modeling to describe the data flow between selected SWSC processes and to decompose these processes into their functional components. This resulted in a SWSC "AS-IS" process definition. Aspects of this model were employed to prepare a proposed SCAI Process Architecture. This $IDEF_0$ depiction of the SCAI project process model will be augmented through the use of ETVX based modeling techniques to depict a more control oriented view of the process activities.

Using both process techniques to explain process concepts to the SCAI project personnel and management, the data collected from both views will be consolidated into the current SCAI Process Architecture which addresses both the architectures for the application and Domain Engineering processes. The $IDEF_0$ and the ETVX based techniques are both necessary to gain project consensus for the process. The $IDEF_0$ technique supports the concept of abstracting process components as process service objects and portrayed artifact flow between process service objects. The ETVX view will bring the $IDEF_0$ abstractions into focus by concentrating on describing activity precedence, process navigation, and exception handling. The resultant process work will provide the process architecture to support the SCAI project. This megaprogramming based system development process will also represent the SWSC "TO-BE" process. These processes will be periodically refined, based both on data collected from SCAI project use, and from future SWSC project users.

1. See Appendix B for more information on these templates.

The overall SCAI process approach is shown in Figure 8.

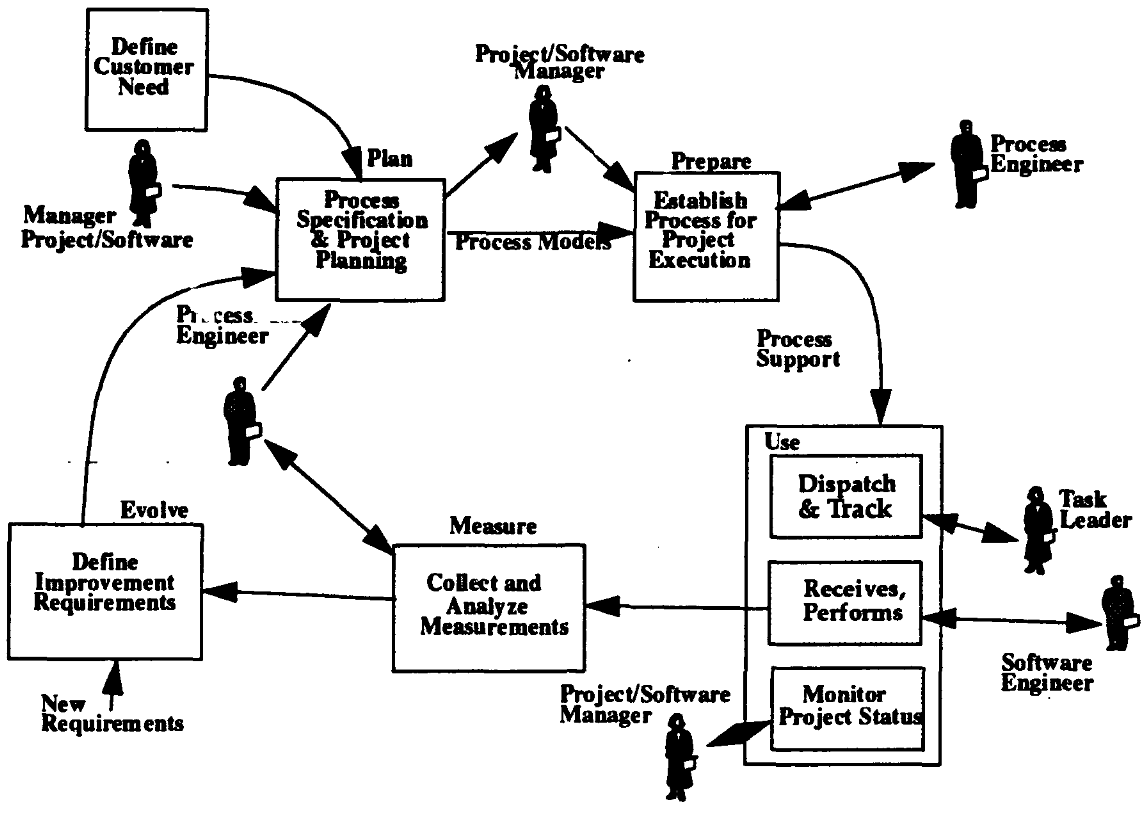


Figure 8. Process Cycle: Definition, Automation, Use, Improvement

Project/Product Context: The analysis and modeling of the SWSC "AS-IS" process along with the unique requirements for the SCAI project will form the basis (or the requirements) for the SCAI process. During this phase, the process and project engineers will become familiar with the processes that are currently in use, as well as the new processes needed to support the RICC technologies and megaprogramming.

Plan: Based on the project and product context established in the above phase, a process specification will be developed using IDEF$_0$ models and Information Organizer Templates. These models and templates will in turn be used to develop detailed SPMS models. Project planning will them be performed by supplementing the SPMS models with resource, schedule, and artifact instance information SPMS will then export all of this information into CAT Compass, the SCAI project management tool. This represents a key feature of the SCAI approach referred to as process-driven project planning.

Prepare: The above activities have established a well defined process and created a project plan for managing that process. The next step is to prepare for automated enactment of the process using the Project CATALYST and Process Weaver capabilities. This step is essentially the encoding and instrumentation of the process to be executed by the Process Weaver engine during project execution.

Use: The steps of the process and individual work tasks will be managed by Process Weaver throughout the SCAI project. In order to support a gradual transition to process driven development and to spread the considerable cost of the process definition and enactment, a phased plan was instituted.

During the Preparation Phase the SWSC "AS-IS" process will be documented using IDEF$_0$. An overall IDEF$_0$ top level Process Architecture of the SCAI will be produced representing the framework of the SWSC "TO-BE" process.

SCAI Release 1 will be performed using manual enactment supported by the IDEF$_0$ models and associated artifact definitions.

SCAI Release 2 will phase-in the use of the automated process planning and support tools. Process improvement will be performed based on the experience from Release 1. A formal process definition will be produced including detailed IDEF$_0$ and SPMS models as well as Information Organizer Templates. Automated metrics capture will be employed based on instrumentation embedded in the processes.

SCAI Release 3 will repeat the process as refined based on the measurements from the previous release.

Measurement: As noted above, measurements taken during the execution of each iteration through the process will be analyzed to determine how well the process performed. Amadeus will act as the primary measurement capture and measurement data base mechanism. SPMS and Project CATALYST will also support measurement capture and will provide the data to Amadeus for incorporation in the measurement data base.

Evolve: Based on the result of analysis from the previous step, a set of process improvement requirements will be defined. These requirements will then act as input to the next cycle of the process planning phase. As this cycle repeats, the SCAI process will continue to improve and by the end of the SCAI project will represent the "TO - BE" process for future development with-in SWSC.

3.2.2 Summary of Accomplishments

The SCAI team accomplished the following during the Preparation Phase:

(1) Created the SWSC "AS-IS" process model using IDEF$_0$.

(2) Created a SWSC "TO-BE" process architecture using IDEF$_0$.

(3) Developed the Information Organizer Templates for formally defining the SCAI process.

3.2.3 Analysis

Much time and effort was spent by project and process engineers learning about all the different methods and approaches being proposed for incorporation into the SCAI megaprogramming software development and maintenance process.

IDEF$_0$ was used to define the SCAI Process Architecture. The following steps were iterated to develop this model:

(1) Gather data by reading literature and interviewing experts.

(2) Bound the subject matter in a context diagram and define the purpose and viewpoint of the model.

(3) Structure activities and interfaces by gradually introducing more detail and refining elements in the model as more knowledge is gained.

(4) Present the model to experts and respond to their comments. [1]

The information in the architecture is examined over time as the model is used to conceptualize how the prevalent SCAI technologies fit together as a whole. Several iterations of the model were developed as the project explored processes such MOIA, Cleanroom, OOA/D and the Ada Process Model with the accompanying technologies. The process architecture defines the Space and Warning System Engineering Meta-Context and the subprocess of the Space and Warning Systems Engineering processes, as well as the SCAI Process Engineering Context.

1. *SCAI Process Architecture IDEF0 Model*, Version 1.0 November 18, 1993, Prepared for: HQ AFSPACECOM Space and Warning Systems Center Directorate of Plans and Programs, Prepared By: SofTech, Inc. 985 Space Center Drive, Suite 320 Colorado Springs, Co 80915 (Currently CACI).

The SEI and STARS conducted a process definition class in December of 1993, focusing project attention on the criteria which are required to define enactable processes. Using the high level SCAI Process Architecture and the Information Organizer Templates project process, definers are expanding the definition of the project processes in the following areas:

(1) System Engineering

- Prepare Specification
- Requirement Model Development
- Architecture Model Development

(2) Incremental Software Development

- Mission Application Development
- Database Development
- Display Development
- Message Development

(3) Configuration Management

(4) Project Metrics

(5) System Certification

These processes will be generalized after the initial definitions are complete, and used to form a set of Domain Engineering processes.

The report on SCAI experience and lessons learned for this process definition expansion, including the use of the templates, will be provided in the 1994 update to this Experience Report. The lessons associate with developing the Information Organizer Templates are found in the STARS/SEI Technology Transition Experience Report. [1]

3.2.4 Lessons-Learned

Converging on Technical Approach

Lesson: How technology is transitioned between organization is as important as the technologies that are transitioned.

Technology transition is at least N directional with N being dependent on the number of organizations that are advocating the use of a technology. When you transfer technology into an organization both the transmitter and the receiver learn from each other. We spent a lot of time learning about technologies before we were able to integrate them into an approach. This was true for developing an approach to define our processes as well as for developing a process for Domain/Application Engineering.

Developing IDEF$_0$ Representation of the SCAI Process Architecture

Lesson: IDEF$_0$ served as a good medium for representing SCAI Process Architecture.

The Design/IDEF tool being used for SCAI works well for SCAI purposes of depicting the Process Architecture. Its primary strength is that it does a good job of graphically depicting the context of the SCAI engineering process relative to other SWSC processes that were modeled using IDEF$_0$ as part of the CIM

1. Linda Parker Gates, Richard W. Phillips, STARS/SEI Technology Transition Experience Report November 30, 1993, Software Engineering Institute, Carnegie Mellon University Pittsburgh, Pennsylvania

initiative. Other modeling technologies are being used to supplement this process definition with information to facilitate process-driven planning, process enactment and measurement of the processes and the artifacts produced by them.

Lesson: Process technologies can be combined to meet the differing needs or objectives of an organization. A single technology does not capture all the process information required.

The IDEF methodology was not fully utilized by all process team members. This may be due to the fact that in the area of process definition, there were multiple technologies being proposed as appropriate for the SCAI project. No formal training in the alternative technologies was provided to all process team members early in the process definition life-cycle, so it took a while for the team to develop a strategy as to how the technologies would play together to create a process driven organization. IDEF gained wider acceptance near the completion of the process architecture development.

Lesson: Review of the Process Architecture was painful.

IDEF methodology calls for a Kit review where the model author provides reviewers with a drafted package, on which the reviewer is to place written comments, which the author then responds to. This package is passed back and forth until the model is complete and recommended for publication. We tried to use this approach but given constraints on team members time, authors did not receive many comments. The authors also tried using unstructured interviews and walkthroughs to review the model.

Lesson: People review process architecture when they are ready to use it.

What we noticed was that people did not really critically review the Process Architecture until they actually started to use it to perform the process, or to develop the lower levels of process definition.

Lesson: Review of high level process architecture by key people is critical to the minimization of rework.

It is extremely important to ensure that close attention is paid to ensuring that the high level process architecture is reviewed in detail by key personnel. This is vital so that process definers can more efficiently and accurately expand the definition of project processes.

3.2.5 Recommendations

Technology: When you plan to integrate multiple process definition technologies, any time spent by the team identifying and co-developing the process definition objectives is time well spent. Team members should also become familiar with all the technologies being considered or planned for use, and the functions each can or will provide.

Review Process: Conduct the review of the high level process architecture in a round table discussion format. Simply passing out copies and soliciting feedback did not produce the necessary results. Although sometimes time-consuming, the round table review discussions proved much more effective.

Pilot: Perform a pilot evaluation of process automation technology prior to operational use. Automated enactment of processes to drive project planning and the dispatching and tracking of a tasks status is only very recently being prototyped. Manual enactment of defined processes has not really been tested and it is unclear whether this would be more effective than the automated dispatching of tasks and the automated tracking of task progress. Earlier prototyping should have been promoted even if it might have meant a certain amount of re-work due to evolving PSS technologies.

4.0 SEE Support

4.1 Introduction

Megaprogramming, according to STARS, requires a significant level of Software Engineering Environment (SEE) support to provide automated assistance to the product-line's process definition and enactment. During the Preparation Phase, the Loral/STARS team continued work on a process-support tool set that is intended for use during the Performance Phase. In the absence of full definition of the SCAI process, it was not possible to finalize the SEE configuration. Certain assumptions about the process were possible, however, allowing the team to install an initial increment of SEE workstations and to populate them with software tools to support key process activities. In addition, a preliminary SEE integration strategy was established, centered around AIX Workbench as the underlying tool-integration framework and the STARS-sponsored process-support tool suite as the basis for process control integration.

Long-term Objectives

The requirement for the SCAI SEE environment is to create a framework populated with tools, supported by well defined processes and procedures, tailored as necessary to the needs of the particular application domain. In addition, the environment shall provide well defined software engineering, project management and life-cycle control procedures, processes and methods.

4.2 Preparation Phase Analysis

4.2.1 Plans

Preparation Phase Objectives

To prepare to address the long-term objectives, the major Preparation Phase objectives were to:

(1) Acquire and install the first increments of SEE hardware and software and integrate the new assets into the existing SWSC/ SMX SEE network, and

(2) Formulate a joint SEE integration strategy with the Air Force and begin the integration work in preparation for the Performance Phase.

The intent of the Preparation Phase was to build up sufficient SEE capability to support the project's initial Performance Phase activities.

Assumptions

Certain givens were assumed at the outset of the project.

(1) The products in the SEE must:

- Follow open integration standards and
- Be available on several hardware platforms.

(2) The SCAI SEE needs to support all members of the SCAI development team, which was initially planned to be about 20 engineers. Adding management, support, and Loral STARS personnel would increase the number of users to over 30.

(3) The application will include about 200 KSLOC of Ada, with major portions of the application being reused from existing work.

(4) The SCAI SEE would support the following process areas:

- Product-line management
- Domain Engineering
- Application Engineering
- Application target testing

(5) Some use would be made of AF residual assets, including hardware, software licenses, and network elements[1].

(6) The majority of the SCAI SEE elements would be commercially available products.

(7) Some of the SCAI SEE elements would be products under development or in beta test. This is especially applicable to products coming from STARS technology development efforts.

(8) A major effort will be the integration of the SEE to the highest level possible, based on the capabilities of the underlying integration mechanisms, and the intended use of the tools on the Demonstration Project. The creation of the SCAI SEEs is limited by several specific constraints.

- Budget - The FY 1993 budget for the SCAI SEE was set at $500,000, with acquisition starting in January 1993 and completed by July 1993. The objective was to spend about half of this budget on hardware and about half on software. FY 1994 will require a similar budget, with a plan of spending 30% on hardware, 40% on software and services, and 30% on training[2]. Subsequent year budgets will have to address upgrades, maintenance, and potential staff growth.
- Security - A portion of the project artifacts will be DoD classified, requiring a segment of the SEE to be classified with no outbound external communications. This security situation has had a significant effect on the connectivity assumption.
- Work Locations - Some members of the SCAI engineering team will be located in separate contractor facilities, and do not have access to the SCAI SEE.
- SEE Product Development - Some of the elements of the SCAI SEE are not fully developed and are not yet commercially available.(SPMS and Project Catalyst are not fully developed).

Planned Activities

Loral FS, as the STARS prime contractor paired with the Air Force, took on the role of coordinating the initial acquisition for the Preparation Phase. The intent was to transition this capability to the Air Force for increments following the Preparation Phase. Responsibility for acquiring and installing some incumbent products (including the Rational Ada support environment, UNAS/SALE, etc.), rested with the Air Force.

The Air Force retained installation and system management responsibility, in keeping with their existing role for other SEE assets.

A joint Air Force/Loral SEE working group was established for coordinating the team's SEE needs and developing solutions and schedules. The overall SEE integration strategy was tied to the SCAI process; and since the process was being developed during the Preparation Phase, coordination with the process support group was required.

Loral FS was to continue development and begin installation and training for the STARS-sponsored process support toolset.

1. *Demonstration Project Baseline Report* Loral STARS CDRL No. 5052, April 1993.
2. *SEE Integration Plan*, Loral STARS CDRL No. 5200, June 1993.

4.2.2 Summary of Accomplishments

(1) Acquired and installed initial increments of SEE hardware and software;

(2) Developed initial overall SEE integration strategy, documented in Version 2.0 of the SCAI Demonstration Project Management Plan (SDPMP);

(3) Enhanced RICC Architectural Infrastructure support software (refer to Appendix B for details) and continued movement towards commercialization; and

(4) Continued development of STARS-sponsored process support toolset, including integration work to establish a coherent process support system; conducted early pilot work with Software Process Management System (SPMS), one of the key tools in the toolset.

4.2.3 Analysis

Early in 1993, a joint team of Loral STARS and AF engineers was established to handle the SCAI SEE. Plans and schedules were established. This effort was described in a series of documents during 1993, which detail the accomplishments and changes in the plan[1].

In creating the SCAI SEE, the joint team acted as a general contractor. They completed selection and installation in stages, as decisions were made, and products arrived. The team members were the consensus gatherers and purchasing agents.

The AF team members were the installers and initial users. The team was supported by several product vendors.

While they made significant progress against the plan, several activities were not completed or reached only limited success. In general, most activities were far more time consuming and complicated than initially envisioned. There were many factors and options that needed consensus decisions. Many of the issues were unavoidable, and there are no recommendations on how to eliminate them, except to plan more labor. The following is a list of some of the issues that impacted the results of the establishment of the SCAI SEE.

(1) The AF and their contractors had an installed base of software and experienced users in that base. The selection of the SCAI SEE products required significant consensus efforts to incorporate this existing base.

(2) The joint Loral/AF team worked in different locations and different time zones. Phone line and electronic communications (E-Mail and FTP) into Peterson AFB were limited. This situation placed a stress on normal communications between team members.

(3) The SEE included products from many vendors, integrated through several open system standard interfaces.

(4) Several of the products (process support tools) had release and development delays. The installation and use of the products were less mature than planned.

(5) Products and prices changed during the selection and acquisition period. Budgets and plans required several adjustments.

(6) The SEE was distributed between Loral, the AF, and AF contractors in several locations. This issue required effort to establish network capabilities and diverted attention from integration efforts.

1. *SEE Integration Plan*, Loral STARS CDRL No. 5200, June 1993.*Demonstration Project Management Plan*, Version 1.1, Loral STARS CDRL No. 5050, June 1993, *SCAI SEE Description*, Loral STARS CDRL No. A010, October 1993, and the *AF/STARS Demonstration Project Management Plan*, Loral STARS CDRL No. A009, October 1993.

(7) A major portion of the SEE is classified, and without connections off the AF base. This was not in the original scope of the STARS Demonstration Project plan. This issue limited support from people outside the classified SEE.

(8) Less than half the planned funding was available in 1993. This delay in funding left the acquisition of classes of products to 1994, and reduced the scope and function of the SCAI SEE in 1993.

SEE Strategy Development Experience

The effective use of megaprogramming concepts requires advanced technology support. Methods and tools are required to achieve compatibility of data, control, and processes, directed at developing shared assumptions about the system being developed.[1]

The long term requirements for the Demonstration Project SEE are stated in the "STARS Follow-on Contract".[2] The following paragraphs provide the relevant extracts of that contract:

> "In general terms, the environment shall consist of a framework populated with tools, supported by well defined processes and procedures tailored, as necessary, to the needs of the particular application domain. In addition, the environment shall provide well defined software engineering, project management and life-cycle control procedures, processes and methods.
>
> The contractor (Loral) shall develop and deliver to the Demonstration Project team, a product-quality, multiple seat software engineering environment tailored to a the specific application domain in accordance with the requirements of the Demonstration Project. The environment will serve as the vehicle for the developing and demonstrating this technology and for supporting new software engineering procedures, processes, and methods.
>
> The contractor (Loral) shall, using the results of previous STARS activities, install and demonstrate the software engineering environment tailored to the specific domain of the Demonstration Project."

The SCAI SEE strategy to meet the SEE requirements is based on the following set of variables.

(1) connectivity,

(2) number of users,

(3) user roles, responsibilities, and tasks,

(4) SCAI team process definitions,

(5) artifact sharing,

(6) user experience,

(7) cost/benefit of integration effort, and

(8) technology transition value to SWSC support team.

Integration of the SCAI SEE is grouped into four areas, the Presentation Integration, Process/Control Integration, Tools, and Data Integration. Cap Gemini's Process Weaver, IBM's WorkBench, and standard relational databases are the primary mechanisms of integration in the SCAI SEE. The following Figure 9 illustrates the integration areas and components.

1. Boehm, B. W., *Megaprogramming*, Preliminary Version, April 1991.
2. *STARS Follow-on Contract*, F19628-93-C-0129, ESC USAF, Mascon, August 1993.

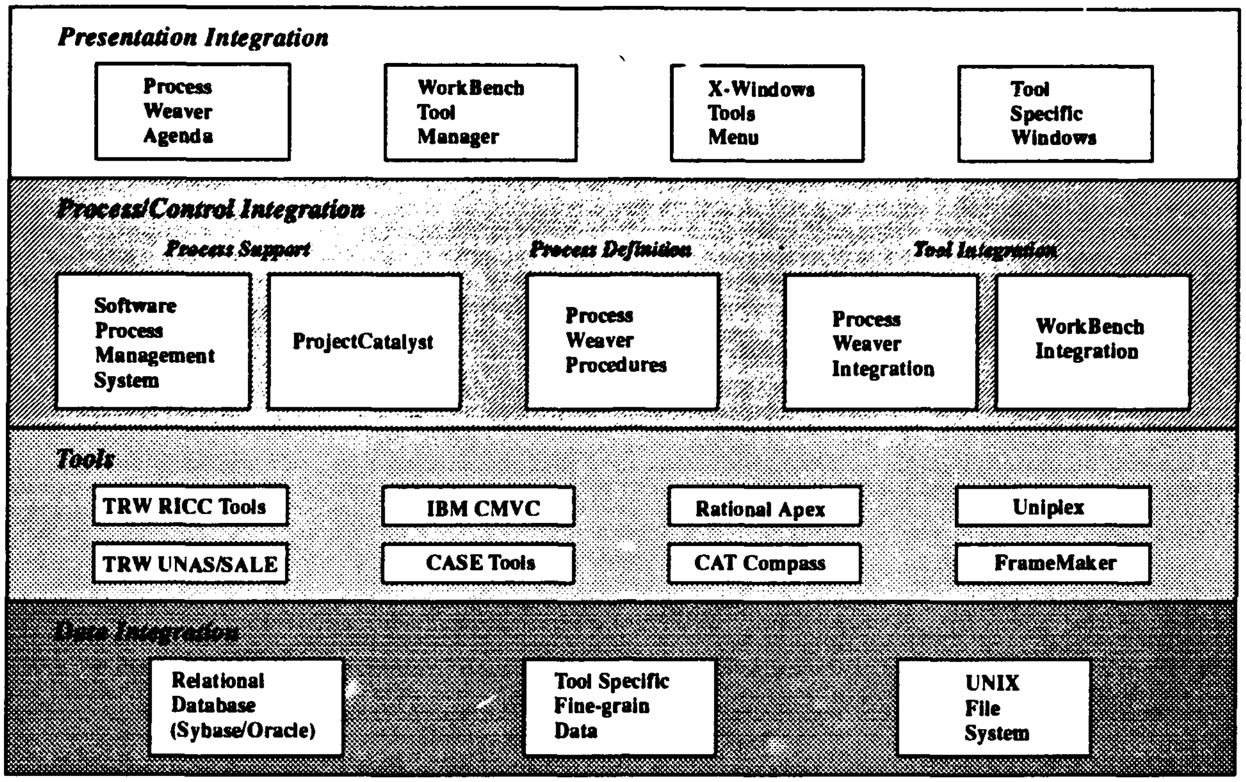


Figure 9. Integration Areas

The main features of the Presentation Integration area are the project process guidance provided by the Process Weaver Agenda and Work Context windows. The WorkBench Tool Manager provides an interface to tools outside of project processes, such as mail. Tools are invoked through Process Weaver and WorkBench, and display their operation specific windows. The X Windows tools menu is used to start Process Weaver and WorkBench.

The Control/Process Integration area includes the Process Weaver and WorkBench integration mechanisms. Process Weaver procedures for the defined project processes provide the majority of process guidance in the SCAI SEE. The STARS developed Software Process Management System (SPMS) and ProjectCatalyst help model, plan, and define the project processes to generate the Process Weaver procedures.

The Tools area includes tools selected by the SCAI team, based on their currently installed products and the skills of the engineers.

The Data Integration area is supported by commercial relational databases, the operating system file system, and tool specific fine-grained data storage.

SEE Process Experience

To meet these goals, within the constraints and assumptions, an iterative and incremental approach was followed. The following is a list of the major elements of the process and the activities within each element. The observations, analysis, and recommendations in this report are against this approach.

(1) Select SEE Products: research products, gather requirements, define criteria for selection and publicize results of selection

(2) Acquire SEE Products: develop preliminary plan for acquisition and use, justify cost to management, initiate acquisition, provide technical support for acquisition and verify receipt of products

(3) Assemble SEE Products: install products, run sanity tests on products and adjust products for local installation specifics

(4) Integrate SEE Products into a SCAI SEE: develop experience in using product, integrate products using process, presentation, control, and data integration techniques, publish administrator guides and publish user guides

(5) Deploy the SCAI SEE: demonstrate the SEE to users and release the SEE for general use

(6) Revise and Maintain the SCAI SEE: collect problem reports, respond to reports, collect change requests, operate change board and survey users

4.2.4 Lessons-Learned

The following are some of the initial lessons relating to the SCAI SEE efforts. There are other lessons that are emerging from observations, but they have not been fully analyzed and are not ready for reporting. They will be documented in future versions of this experience report.

Lesson: The agreements and licenses for acquisition and maintenance are widely different for each product.

This situation creates variation in levels of support and installation flexibility. For some products, there is sufficient immediate support, while for others there is virtually no support. Some products are bound to specific system configurations, and it is time consuming and sometimes costly to go to the vendor for adjustments.

Lesson: It was difficult to plan when a product would arrive or how it would integrate into the SEE.

The variability of products being delivered to the SCAI SEE assembly site requires frequent changes to schedules and plans. When trying to integrate several products, the delay in one product will delay in the whole integration. Equally serious is the lack or limitation of intended function due to product development problems. Usually adjustments can be made for these problems because of the variety of capabilities in the other products and the use of manual processes, but disruptions in the plans will occur.

Lesson: Upgrades in the operating system (AIX) has a significant impact in our integration efforts.

There is often significant delays in upgrades of tools in the newer operating systems.

Observation: Integration complexity increases as the number of tools/vendors increases.

There are clearly advantages and disadvantages to having a SEE which requires integration of products from multiple vendors. The SEE Team is in the process of distinguishing these factors and is trying to determine how the factors impact integration complexity.

4.2.5 Recommendations

Based on the AF/STARS Demonstration Project SEE experiences, the following general planning guidelines should be followed.

Incremental Integration: Plan an incremental approach to integration with few critical dependencies.

Expertise/Training: Plan for concentrated training of the integrators or hire product experts to overcome the experience deficit. Do not select the products that the SEE integrators know, and lose the advantage of existing project personnel experience.

Staffing Requirements: When selecting, acquiring, and assembling a SEE for a small software engineering team (about 20 engineers), plan about 16 labor months for the effort, with a minimum elapsed time of about six months. This does not include the integration, deployment, or support efforts. If delays occur in this effort, as experienced on the AF/STARS Demonstration Project, additional labor months will be required. The AF/STARS Demonstration Project, spent an estimated 14 labor months on these activities during 1993. During 1994, to complete these activities, the AF/STARS Demonstration Project will spend another 8 labor months.

Prerequisites to Acquisition: The selection or acquisition approach should not be started until you have a general definition of the scope of the engineering effort and the processes that the engineers will follow. The recommendation is to not assign anyone on a proposal team to SEE product definition, until the software engineering process has a high level definition and until the SEE product evaluation process is defined and documented. It is estimated that because of the lead time and cost needed to acquire SEE products, most projects define a tool set early in the proposal, and get the SEE products locked into the contract cost. The definition of processes is viewed as something that can be done after the contract award. Unfortunately, this leads to parts of the process with no automation support, or products being acquired without a process to use them (shelfware).

Operating System Releases: Plan integration of SEE Products around the operating system upgrades. Operating System upgrades severely impact the integration efforts.

5.0 Metrics

5.1 Introduction

Metrics are instrumental in providing information to quantify software life cycle cost, quality, and capability differences resulting from the presence and absence of megaprogramming technologies. The SCAI project will quantitatively and qualitatively measure the effects of megaprogramming on the development effort and the resulting product. During the Preparation Phase, the SCAI team formulated a preliminary metrics process. The foundation for the metrics process is the formally defined set of project goals. The team defined the project goals (refer to Section 1.2, Vision, Mission and Goals) and began detailing the metrics collection and analysis approach.

Long term Objectives

The goal of the SCAI Project metrics activity is to establish a measurement process along with the necessary organizational and technological support to:

(1) Enable the evaluation of progress toward achieving the goals of the SCAI project,

(2) Support continuous improvement of the processes being applied, and

(3) Provide a historical artifact to be used by future projects in determining the potential benefits of applying the SCAI megaprogramming approach.

5.2 Preparation Phase Analysis

5.2.1 Plans

Preparation Phase Objectives

During the Preparation Phase the STARS project metric objectives were to:

(1) Define metrics process (approach).

(2) Gain buy-in on SCAI goals from project personnel.

(3) Prepare for long term objective one above:

- Document initial set of measurements.
- Begin capturing some data.

Assumptions

The following assumptions were made at the start of the metrics effort:

(1) Initial guidelines and methods would come from:

- DoD Core Measurements.
- IDA 12/92 prototype Measurement Plan.
- Goal/Question/Metric (GQM) Method.

(2) Some staffing would be available during the Preparation Phase (1 Full Time Equivalent person of effort from a 3 person team).

Planned Activities

The Metrics Team decided early in the Preparation Phase to combine the metrics work with the project's overall goal-setting work, with the rationale that the primary basis of any measurement activity is to help determine how well the goals are being achieved. During the Preparation Phase, the team evolved the following process description to depict the integrated goals and metrics activities.

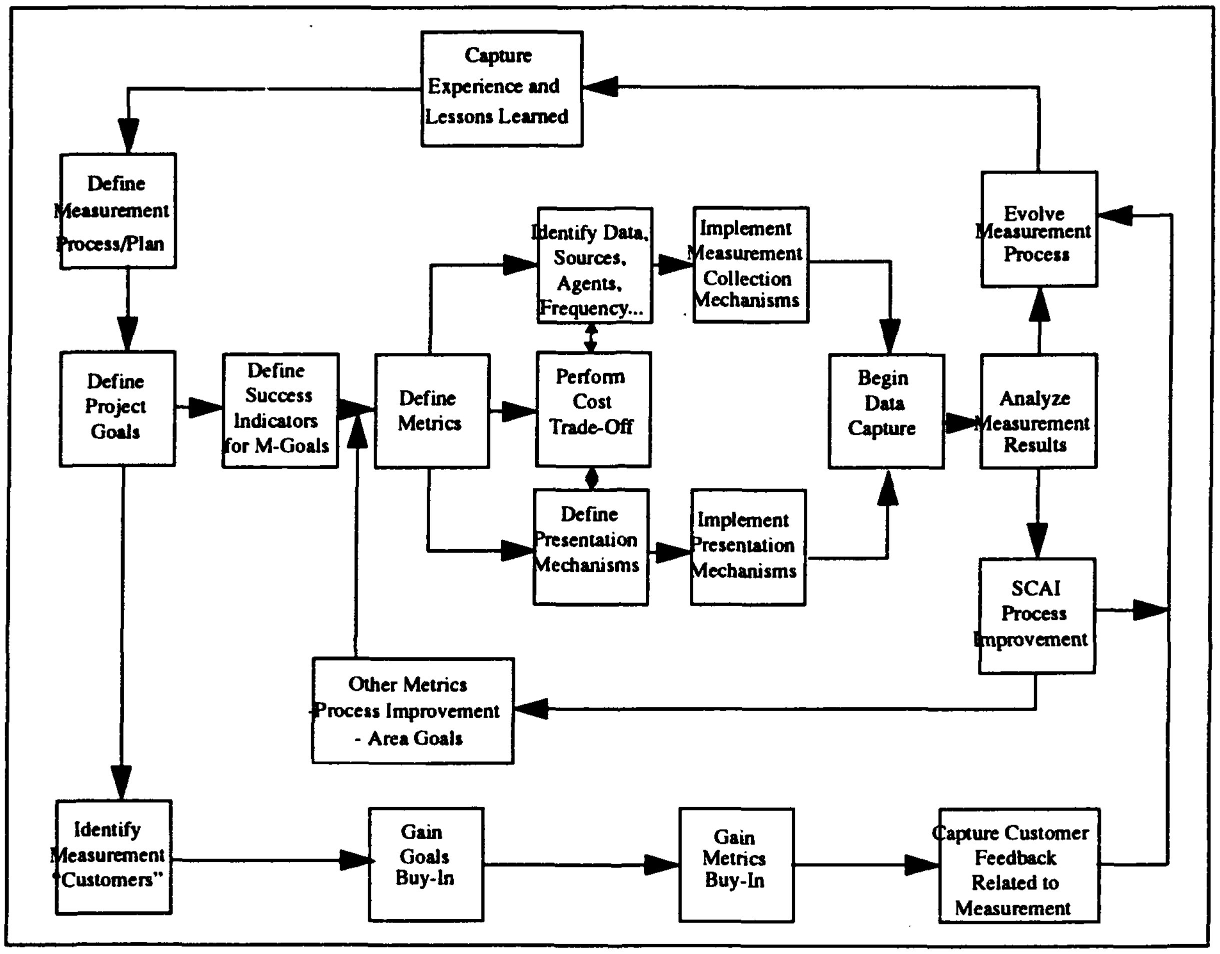

Capture Experience and Lessons Learned
Define Measurement Process/Plan
Define Project Goals
Define Success Indicators for M-Goals
Define Metrics
Identify Data, Sources, Agents, Frequency...
Implement Measurement Collection Mechanisms
Perform Cost Trade-Off
Define Presentation Mechanisms
Implement Presentation Mechanisms
Begin Data Capture
Evolve Measurement Process
Analyze Measurement Results
SCAI Process Improvement
Other Metrics Process Improvement - Area Goals
Identify Measurement "Customers"
Gain Goals Buy-In
Gain Metrics Buy-In
Capture Customer Feedback Related to Measurement

Figure 10. Metrics Process

The measurement process as shown in Figure 10 includes:

(1) Selection of SCAI Project top level goals.

(2) Identifying a set of success indicators associated with each goal.

(3) Defining a set of metrics which could verify the successful achievement of the goal.

(4) Establishing an approach for capture and collection of the metrics.

(5) Defining an approach to presenting and analyzing the measurement results.

(6) Instituting a continuous improvement cycle for the measurement process itself.

The approach to metrics definition is both top down and bottom up. From a top down perspective the project goals will be defined and refined. The purpose of the metrics activity is to determine if the goals are being reached. The goals provide the rationale for the actual measurements collected. You could also think of the goals as providing requirements on the metrics activity.

From a bottom up perspective, however, there is a need to determine what specific data can be gathered and how/where it will be captured. The guidance and templates provided by the DoD Core Measurements are used to help define these specific details.

A cost benefit trade-off will be performed to determine what measurements are practical. It is at this point that the top down and bottom up approach meet. The result will be an affordable metrics approach that provides a tool for project management, process improvement and future decision making regarding the application of megaprogramming.

Also shown in Figure 10 is the need to identify and gain buy-in from the "customers" the metrics processes are being performed for. These customers include project management from the SWSC, STARS and ESC as well as all the SCAI project participants.

A key element of all aspects of this metrics process is the consensus building approach associated with each of the activities. The process of defining goals and success measurement can be very effective in team building and flushing out the overall project process. To this end, each aspect of this approach will be developed with the involvement of all project personnel. Each iteration of the process will be developed using interactive group techniques to assure group acceptance.

5.2.2 Summary of Accomplishments

(1) The SCAI Metrics Plan was published in October.

(2) Process (approach) was outlined and initiated.

(3) Version 2 of the Goals were documented and reviewed by project.

(4) Version 1 of the success indicators for 2 of the 3 top level goals was documented and reviewed by project. Goal 3 (Technology Transition) deferred until mid ways through the project.

(5) The initial draft of desired measurements produced covering about 50% of the needed measurements. Many specific details need to be defined prior to capture of these measurements. Much of this detail will eventually be part of the SCAI processes.

5.2.3 Analysis

For each of the five points noted here, the specific lessons learned are described in Section 5.2.4.

We went in search of the SCAI processes and couldn't find them documented, so we defined and documented what we know we want to measure, and we still need to define how we will gather measurements.

(1) We were able to refine Goals/Subgoals and Success Indicators by meeting in small focus groups of 2-5 interested people for 3 hour meetings.

(2) We have taken the Goals/Question/Metrics (GQM) approach and adjusted it to construe the questions as success indicators.

(3) We established collection categories, which appears to make the metric derivation and implementation approach more manageable.

(4) We were able to start this process with a small team of people and a reasonable amount of effort (1 FTE), to get an initial understanding of measurements needed on the project.

5.2.4 Lessons-Learned

Lesson: The level of process definition and the precision of metrics are interdependent.

We would have wanted to say "Make sure processes to be measured are well specified," prior to defining metrics to be captured. However, what we really think is possible is that an organization can:

(1) Let the metrics support the process definition activities. The need to collect key pieces of information often highlights critical points in the process which need attention.

(2) Have more human interaction when collecting measurement definition data. Even if the process is not completely defined, humans can often provide the desired measurement data which at some later date might be captured automatically based on a specific process definition.

(3) At some point stop metric definition activities and put more effort towards process definition. There is a trade-off to be made here. If your metrics definition gets too far ahead of the process definition, the result may become redundant, an inefficient use of resources. It might be better to move resources to the process definition activity and then pick up the metrics implementation when the process has stabilized.

Lesson: Goals/Success Indicators are critical and need to have consensus.

It is never too early to define goals. Everyone on a project assumes they understand the goals. Unfortunately, put these people in a room and start discussing the goals and you find that there is a great deal of confusion and disagreement.

The process of defining goals leads to team and consensus building. Much of the initial metrics work related to defining goals and success indicators should be carried out regardless of any needs related to metrics collection. We used a technique of inviting small groups of project personnel to multiple half-day (about 3 to 3.5 hours) meetings on a subset of the project goals. The subset of goals and the individuals invited to the meetings were matched up so that the specific goal was critical to the activities of the group.

The approach worked well. There was sufficient time for detailed, and often frank, discussions to take place. We held two rounds of meetings (5 different groups in each round). The first round concentrated on goals and the second round refined the goals into success indicators. We captured issues from the first round in order to avoid getting too detailed and too hung up on definition. We found that by the second round most of the issues had resolved themselves. What was happening was that the goals meeting were a catalyst to forcing discussions on important topics.

The risk we accepted with this approach was that not all project personnel were able to participate in the detailed discussions on all the goals. It is critical to do a careful job of matching people to goals for this approach to succeed.

Lesson: Success indicators/questions are effective for refining goals and goal definition.

The questions (from the GQM method) were worded such that a positive answer would provide an indication that we may be successfully achieving the goal. This gave the questions a very specific focus and helped to better clarify the goal.

Lesson: The approach tends to explode, so the team needs to work constantly to aggregate.

We first defined 3 top level goals, then sub goals and sub-sub-goals. The success indicators continued this expansion. It became clear that we needed to constantly try to aggregate the information and not let this expansion become unmanageable. We knew that the measurement definition would eventually pull things back together. But the concern was that the process of expansion would get out of hand and you would never be able to trace all the goals and success indicators back to the specific measurements.

Lesson: Collection categories made the process more manageable.

We noted during the process that there would be four basic approaches to capturing the measurement data:

(1) Automatically via the SEE.

(2) Extracted from financial systems (staff hours/cost).

(3) Manual inspection of work products.

(4) Surveys of Project personnel and potential SCAI users.

This made the organizing of the measurements much easier and allowed us to divide up the work of implementing the collection mechanisms.

Lesson: A significant amount of effort is required for metrics definition and collection, and dependence on the project infrastructure and process definition is clearly evident.

We were able to do the following activities with a group of three people spending approximately 1/3 of their time each on:

(1) Defining a metrics plan/process,

(2) Coordinating the refinement and consensus building for the goals and success indicators, and

(3) Developing a first draft list of the needed measurements and mapping them to the success indicators.

The effort was spread over a 5 month period, May through September (total of 5 person months). However, at this point the workload increases significantly. The activities to perform next are: defining specific measurements, data values, frequencies for collection, mechanisms for collection, database capture definitions, and presentation formats. The level of effort for this activity is estimated at 2.5 persons.

5.2.5 Recommendations

Value of Metrics: Emphasize that the value of metrics to the team or team members is the ability to distinguish:

(1) what they intend to accomplish and

(2) what they are actually accomplishing.

With visibility into what has been accomplished, given what one intends to accomplish, team members have the ability to take actions that can actually address getting back on course when deviations occur. However, in order to be able to accurately report what has been accomplished, the atmosphere within the organization needs to be such that individuals and teams do not fear retribution for being off course.

Team Exploration: During the Preparation Phase of your project, you explore with your team:

(1) How they relate to or feel about measurements?

(2) If they were sure that the results of measurement would not be used as a basis of retribution, what possible value might visible metrics contribute to their job performance?

(3) What if anything is missing in the organization for people to feel confident that measurement results will not be used against them?

Early Project Focus: Help the project focus on what they intend to accomplish as early as possible, and have them identify how they would know if they have accomplished their intent. This focus is important not only for all project goals, but also for any goals associated with project phases. Once you have identified where people or the team wants to go, give them continuous visibility, through displays, into where they are.

6.0 Conclusion

This report has provided an overview of the SCAI project approach and the experience gained during the Preparation Phase (October 1992 through October 1993). The report will be updated at the end of 1994, mid-way through the Performance Phase, and again in early 1996, at the completion of the project. Interim updates on the status of all the STARS Demonstration Projects will be provided in the STARS newsletter.

We hope this information is of value to your organization, and we welcome any comments.

Appendix A

Acronyms and Definitions

Acronyms

AAM	Application Architecture Model
ABC	Activity Based Costing
AE	Application Engineering
AF	Air Force
AFSPC	Air Force Space Command
AFB	Air Force Base
AIX	Advanced Interactive eXecutive (IBM operating system)
AJPO	Ada Joint Program Office
AMS	Assets Management System
API	Application Programming Interface
APSE	Ada Programming Support Environment
ARM	Application Requirements Model
ARPA	Advanced Research Projects Agency
BRIP	Business Process Improvement Program
C^2	Command and Control
C^2AI	Command and Control Architecture Infrastructure
C^3	Command, Control and Communications
C^3I	Command, Control, Communications and Intelligence
CCPDS-R	Command Center Processing and Display System Replacement
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CFRP	Conceptual Framework for Reuse Processes
CIM	Corporate Information Management Center for Information Management
CINC	Commander in Chief
CM	Configuration Management
CMAFB	Cheyenne Mountain Air Force Base
CMAH	CINC Mobile Alternative Headquarters
CMC	Cheyenne Mountain Complex
CMM	Capability Maturity Model
CMVC	Configuration Management Version Control
COSE	Common Open System Environment
COTS	Commercial-Off-The-Shelf
CSC	Computer Systems Components

CSCI	Computer Software Configuration Item
DA	Domain Analysis
DAM	Domain Architecture Model
DAPM	Domain Analysis Process Model
DASD	Deputy Assistant Secretary of Defense
DBMS	Data Base Management System
DE	Domain Engineering
DID	Data Item Description
DISA	Defense Information System Agency
DE	Domain Engineering
DJAG	Demonstration Joint Activity Group
DoD	Department of Defense
DRM	Domain Requirements Model
ESC	Electronic Systems Center
ESIP	Embedded Computer Resource Support Improvement Program
ETVX	Entry, Task, Validation, Exit
FEA	Functional Economic Analysis
FFRDC	Federally Funded Research and Development Center
FPI	Functional Process Improvement
FSC	Federal Systems Company
FTE	Full Time Equivalent
GAC	Generic Application Controller
GQM	Goal/Question/Metric
IBM	International Business Machine
ICAM	Integrate Computer Aided Manufacturing
IDA	Institute for Defense Analyses
IDEF	ICAM Design Engineering Facility
IDEF 0	IDEF -Activity Modeling Technique
IDEF 1x	IDEF - Rule Modeling - Data Model
IEEE	Institute of Electrical and Electronic Engineers
IM	Information Management
IR	Incident Report
IV&V	Independent Validation and Verification
LOC	Lines of Code
MCCC	Mobile Consolidated Command Center
MCCR	Mission Critical Computer Resources
MCCS	Mobile Command and Control System
MOA	Memorandum Of Agreement
MOD	Modification
MOIA	Mission Operation/Information Analysis
MSS	Mission Support Segment

NFS	Network File System
NIST	National Institute of Standards and Technology
NORAD	North American Aerospace Defense Command
OO	Object Oriented
OOA	Object Oriented Analysis
OOD	Object Oriented Design
OJT	On-the-Job Training
PDL	Program Design Language
POSIX	Portable Operating System Interface
PSE	Process Support Environment
PSS	Process Support System
QA	Quality Assurance
RD	Recursive Design
RICC	Reusable Integrated Command Center
SAF/AQK	Security of the Air Force/Acquisition
SALE	Software Architect's Life-Cycle Environment
SAS	Software Architecture Skeleton
SC	Directorate of Communications and Computer Systems
SCAI	Space Command and Control Architectural Infrastructure
SCF	Software Change Form
SDE	Software Development Environment
SDPMP	STARS Demonstration Project Management Plan
SDR	State Date Repository
SEE	Software Engineering Environment
SEI	Software Engineering Institute
SEPG	Software Engineering Process Group
SET	Software Engineering Technology, Inc.
SMQ	Quality Control Directorate
SMX	Plans and Engineering Directorate
SPADOC	Space Defense Operations Center
SPMS	Software Process Management System
SQL	Standard Query Language
SSC	Space Surveillance Center
SRA	Systems Research and Applications Corporation
STARS	Software Technology for Adaptable Reliable Systems
SWSC	Space and Warning Systems Center
TCP/IP	Transmission Control Protocol/Internet Protocol
TW/AA	Tactical Warning/Attack Assessment
UNAS	Universal Network Architecture Services
USSPACECOM	United States Space Command

WBS	Work Breakdown Structure
VADS	Verdix Ada Development System

Definitions

(1) Software Engineering Environment

A "software engineering environment" (SEE) is a complex system including many software tools, hardware components, and network capabilities communicating through standard interfaces. A SEE provides automated technology support for the people developing systems and the processes they use to develop the systems. The integration mechanisms of the Loral STARS SEE are Cap Gemini's Process Weaver, IBM's WorkBench, and standard relational databases.

(2) Process Support Environment

A "process support environment" (PSE) represents the subset of the SEE that provides the infrastructure capabilities for supporting the development and execution of a project's "process support system." The Loral STARS SEE PSE includes Process Weaver, Project Catalyst, Software Process Management System (SPMS), and CAT Compass.

(3) Process Support System

A "process support system" (PSS) is the system of processes, practices, guidelines and methods that guide project personnel in the planning, specification, development and delivery of a target system. A PSS can be an entirely manual system, partially manual and partially automated, and if practical, entirely automated. A project's PSS created by the Loral STARS SEE PSE consists of SPMS process definitions and the SPMS database, Project Catalyst and Process Weaver project process definitions, and the project's management plan.

(4) Architecture

Two uses of the term architecture are appropriate:

- The methods used to construct a system - including the prescribed use of:
  - Components, such as hardware, software, files, and networking interconnections,
  - Interfaces, such as APIs and standard message protocols,
  - Tools, such as code generators to assist in using an API, and
  - Programming language templates.
- The actual as-built structure of a system constructed using the above approach.

  (See also Domain Architecture, Architecture Models)

Further notes on architecture:

- The word Architecture refers to the "solution space" rather than the "problem space".
- Note that in a megaprogramming product-line, architectural commonality is given a high priority - since it is viewed as a prerequisite for large amounts of domain-specific reuse.
- There is a difference between "architecture" and "models of the architecture". The architecture is how the system is built, and we use models to depict various views of the architecture. When working out an architecture, one can use models as communication and analysis vehicles.

- There are numerous valid views of architecture, and many valid modeling approaches to depict those views. When working out an architectural modeling approach, a project must decide which particular views are important - given the domain, the technology, and the people doing the work. The various models will hopefully be consistent - and tool support for maintaining/checking this consistency may well be appropriate.

The following views are candidates for the SCAI:

- Composition - the building blocks approach: subsystems, CSCIs, CSCs. Further, there are both static and dynamic subviews of this.
- Abstraction - distinguishing among layers of abstraction in order to separate concerns and in order to allow people to work on portions of the design independently. The Shlaer-Mellor ideas of Recursive Design (RD) are an example. Static and dynamic subviews may be appropriate here also (e.g., these-called "bridges" in RD that specify how the higher levels interact with the lower levels).
- Construction Methodology - the approach for fabricating the software. This should be worked out in the context of composition and abstraction above, and might be very specific to the product-line. Subviews of this view might be a set of templates of legal Ada constructs, SALE, SAS and GACs, and RICC tools/APIs. Again, note that the various views should relate to each other but it is important to recognize that all of them coexist; they all can be essential to understanding, communication, and engineering.
- Architecture Model

  A set of views of architecture for an application (or a domain) that provides a basis for communication and analysis. Views can include depictions such as:

  - Interfaces and "building block" composition
  - Layers of abstraction, and
  - Construction methodology
- Domain Architecture Model (DAM)

  An Architecture Model for a family of related applications in a megaprogramming product-line; the DAM identifies:

  - Architectural commonalities, and
  - Methods for adapting the common aspects to specific applications in the product-line.
- Application Architecture Model (AAM):

  An Architecture Model for an application in a family of related applications in a megaprogramming product-line.

(5) Domain Terms

[Note: The motivation for attempting to define this set of terms is that there is much confusion in our communication, because there are so many different possible meanings for the term "domain". It seems clear that we should seldom use the term "domain" without some suitable modifier - such as "application domain" or "user interface domain".

- Domain

  A coherent body of technical discourse that has been selected to assist in reasoning about a system or family of systems.

- Application Domain

  A domain that is restricted to the inherent aspects of a problem as distinguished from the aspects that are peripheral or subsidiary (see Service Domain, e.g.). The purpose in clearly defining the boundaries between the Application Domain and other domains is to allow application domain experts to concentrate on those aspects of the problem that they are most interested in and best suited for.

- Service Domain

  A domain that is viewed as subsidiary to a higher domain. The purpose in separating out a service domain is to remove an area of concern from the discourse of the using domain, freeing the analysts of the using domain to concentrate on only the most pertinent aspects. For example, an Application Domain could be distinguished from a Message I/O domain.

- Subdomain

  Usage of this term is discouraged, since its meaning is so prone to misunderstanding!

- Application Domain Family

  A family of related applications being analyzed as a group due to their similar characteristics; the motivation for treating them as a family is to identify and exploit commonalities among the applications so as to maximize "domain-specific" reuse.

  For example, in the SWSC we are interested in the Space and Warning Domain Family.

- Application Domain Subfamily

  A subset of the applications in an Application Domain Family that has been called out because of their strong similarities, allowing focus by specific application experts and giving rise to the opportunity for focused reuse. In a megaprogramming product-line, the system solutions for a Subfamily would take advantage of Domain Family reuse, but would tailor/adapt the Family's assets in a coherent way across the Subfamily - and they would also take advantage of assets common to the Subfamily, but not used in other Subfamilies.

  For example, in the SWSC we are interested in the "Space" Domain Subfamily of the "Space and Warning" Family.

- Product-Line

  A set of applications developed and maintained using a megaprogramming paradigm - i.e., using a process-driven, domain-specific reuse based, technology supported approach. An essential characteristic of a product-line organization is the presence of a focal point "Domain Manager", responsible for a domain family of applications, as well as the resources and processes used to develop and maintain them

  The Domain Manager may have ownership of other applications in the Domain Family that are legacy systems and are not considered part of the product-line. The primary objective of transitioning to a product-line approach is to exploit the commonality among the applications and to realize a high degree of systematic reuse.

- Domain Analysis

  Possible meanings of the term "domain analysis "include:

  The process of systematically analyzing a system or family of systems by dividing the universe of discourse into separate domains with well-defined interfaces. The purposes of such divisions may include: increasing the ability to attain intellectual control, separating out common aspects from unique aspects within a family of systems, and arranging for good encapsulation in system implementations.

The term is used to characterize "problem" analysis; Domain Analysis is intended to provide artifacts that clearly characterize the problem to be solved and that serve to enhance the "solution" process.

Although the definition allows Domain Analysis to take place for a *single* system, this is really just a special case of a *family* of systems. In other words, even while analyzing a single system, the analysis method attempts to focus on the essence of the problem and strip out most implementation considerations. Thus, the analysis work products should be viewed as supporting a family of application solutions.

- Analysis of the inherent aspects of a single application, avoiding implementation-specific aspects (e.g., Ward. Mellor, McManamon/Palmer)
- Analysis that involves separating the application into multiple parts, each of which is subject to delegated analysis by personnel with expertise in specialized disciplines, such as database management or operating systems (e.g., Shlaer/Mellor)
- Analysis of several related applications with the intent of flushing out and exploiting commonalities. (e.g., Prieto-Diaz, Cohen)

(6) Engineering Terms

- Application Engineering

  Engineering (specifying, developing, deploying, and maintaining) a software-based system solution for an application in accordance with a megaprogramming product-line strategy that applies to multiple similar applications. (See also Domain Engineering.)

- Domain Engineering

  Engineering a family of similar applications in accordance with a megaprogramming product-line strategy, providing a common architectural approach, tools, domain models, and other reusable artifacts for the entire family. (See also Application Engineers.)

- Database Engineering

  Team responsible for engineering the common state data used for an application or family of applications. This may involve modeling the data, developing schemas, defining data dictionaries, and designing queries.

- Distributed Processing Requirements

  Allocation of the capabilities of a system to nodes in a network environment and specifications of their interface and performance characteristics.

(7) RICC - Reusable Integrated Command Center

A generic implementation of a Space and Missile Warning Command Center based on a reusable architectural infrastructure and an associated tool set. The first instantiation of the RICC was a pilot in the Missile Warning domain.

- RICC Architectural Infrastructure

An open systems-based set of APIs and Ada run-time components that were first used to build the RICC MW pilot, and which are being used to build the SCAI application. These consist of the Reusable Human Machine Interface (RHMI) for interactive displays, the Generic Message Parser (GMP) for message handling, and the Query Processor (QP) for database operations. The infrastructure also includes the Universal Network Architecture Services (UNAS) for the run-time network executive. (See also RICC and RICC Tools)

- RICC Tools

  A set of interactive tools used to generate code to the RICC Architectural Infrastructure APIs. These consist of Display Builder (DPT - for RHMI), Message Tool (XMT - for GMP), Query Builder Tool (QBT - for QP), and System Architects Life-Cycle Environment (SALE - for UNAS). (See also RICC and RICC Architectural Infrastructure).

Appendix B

Technologies Contributing to SCAI

Background Technologies

The Appendix will cover the technologies being used on the SCAI project that have influenced the project's technical approach. Many state-of-the-art incumbent technologies existed within the SWSC prior to the arrival of STARS technologies. This appendix provides descriptions of the incumbent technologies, the STARS technologies, and a description of a technology for defining processes created as a result of a process affiliate relationship with the Software Engineering Institute (SEI).

(1) Technologies already in use at the SWSC at the start of the Demonstration Project (incumbent)

- Corporate Information Management (CIM) Process Initiative
- Reusable Integrated Command Center (RICC)
- Mission Operation/Information Analysis (MOIA)
- Booch Object Oriented Analysis
- TRW Ada Process Model

(2) STARS Technologies

- STARS Process-Driven Development
- Automated Process Support Tools
  - ProjectCatalyst
  - Process Weaver
  - Software Process Management System (SPMS)
  - CAT Compass
  - Amadeus
- Cleanroom Software Engineering Process
- Domain Analysis Process Model (DAPM)

(3) SCAI/STARS/SEI Process-Driven Technologies

- Information Organizer Templates

Incumbent Technologies

Corporate Information Management (CIM) Process Initiative

The Deputy Assistant Secretary of Defense (DASD) for Information Management (IM) sponsored a Corporate Information Management (CIM) workshop of the processes performed in conducting software maintenance at the SWSC. DASD IM commissioned a Joint Services' CIM Work Group to conduct this workshop and improve the activities associated with the SWSC. The group conducted its modeling efforts in accordance with DOD 8020.1-M, Functional Process Improvement Program, 1 October 1992. Participants included personnel familiar with the operation of the

SWSC, the Integrated Tactical Warning/Attack Assessment (TW/AA) configuration control process, and operations within the Cheyenne Mountain Air Force Base (CMAFB). The Integrated Computer Aided Manufacturing (ICAM) Definition (IDEF) techniques were employed to do the business process models.[1]

The CIM program is aimed at changing the way people work in the DoD so that the DoD operates with cost optimization and performance excellence objectives. To implement the CIM initiative, the DoD has created a Business Process Improvement Program (BPIP) to encourage a consistent application of process improvement principles and techniques across its services and agencies. The objectives of these techniques is to eliminate duplication of functions and the redundancy of business processes and information systems. The BPIP contains general concepts and steps associated with business process improvement and incorporates the development of a business case for evaluating investments.

"The DoD is using business process improvement workshops to identify inefficiencies, poor business practices, and costly non-value added activities. These workshops enable functional managers to identify current problems, establish costs for business activities, propose change alternatives, and implement business improvements in their organizations and business processes."[2]

The key objectives of the BPIP include:

(1) Building a model and establishing cost and performance measures of the baseline to enable the organization to demonstrate improvements.

(2) Identifying and eliminating non-value added activities.

(3) Simplifying, integrating, and streamlining value added activities.

(4) Emphasizing reuse of assets whenever possible.

(5) Automating only after underlying business processes have been cleaned up.

(6) Aligning goals, policies, and procedures within the CIM Integration Architecture (the reference model that guides all information systems implementation activities providing a strategic framework for making decisions that affect the DoD information infrastructure).

(7) Integrating processes, physical assets, organizations, and data as appropriate to gain economies of cumulative volume and limited redundancy.

To achieve these objectives the CIM Function Process Improvement Technology was developed. It is composed of six key procedures which are further decomposed into subordinate procedures. The following identification and definition of the main procedures is followed by an explanation of the SWSC implementation (or planned implementation) of that procedure.

(1) Establish Functional Project Framework

The initiation and planning of the project is performed during this activity. It includes developing business overviews, weighing and selecting objectives, determining opportunity areas, and establishing the project scope.

The scope of the SWSC software maintenance process modeled and posted includes the following: activities performed by the operator/day staff, creation and coordination of Incident Reports (IRs) and Modifications (MOD) Software Change Forms (SCFs), SWSC planning baseline control, engineering and daily support, and the Integrated TW/AA Configuration Management Process.

(2) Document and Analyze Current Baseline

1. Space and Warning Systems Center (SWSC), Corporate Information Management (CIM) Team, Section 1 Introduction, *Business Process Improvement Analysis of SWSC Software Maintenance. Activity Based Costing (ABC) Foundation Workshop*, Systems Research and Applications Corporation (SRA) at Peterson AFB, Colorado Springs, CO, pg 1-1.
2. D. Appleton Company, Inc., Business Process Improvement: The CIM Initiative, *Corporate Information Management Process Improvement Methodology*, D. Appleton Company, Inc. Second Edition, (3028 Javier Road, Suite 400 Fairfax, VA 22031 (703) 573-7644 1993) pg 6.

During this activity the current business process including the cost of each individual activity is identified. IDEF$_0$ an activity modeling technique developed by the United States Air Force, has been mandated by the CIM Information Technology Policy Board as the mandatory technique to use for modeling the business processes.

The IDEF$_0$ models are then used to drive the Activity Based Costing (ABC) process. The ABC techniques reorganize traditional financial information to show functional managers what they do with money, rather than how they spend it. It helps characterize the value of, or need for, each activity.

For the SWSC, the AS-IS process defines the software maintenance process beginning with the recognition of a problem or definition of a desired change and continues through the implementation of a vertical release. Included are the activities associated with processing SCF's through the Integrated TW/AA configuration control process.

(3) Perform Business Improvement Analysis

During this activity the team determines the applicable best business practice for improvement opportunities to the current business process and defines alternative business processes with associated cost and risk for each. Data models for the alternative processes are developed, and alternatives are evaluated for cost benefit to the baseline. The most cost-effective alternative is selected and recommend as the TO-BE business process alternative.

For the SWSC Software Maintenance Business Process Improvement Analysis, the processes being developed on the SCAI Demonstration Project are being used as the basis for creating the SWSC TO-BE activity model. It is anticipated that the SCAI process will eventually replace the current process. The actual migration of the SWSC to the SCAI megaprogramming development strategy is anticipated to require a 5 to 10 year commitment.

(4) Develop Management Plan and Functional Economic Analysis (FEA)

The activities during this process define the management decision and documentation processes. A Functional Economic Analysis is completed on the technical alternatives and presented for management decision. The function portion of this analysis helps people understand what an organization does, how well it accomplished its mission, and how its processes can be improved by creating a picture of how the business is run on a day-to-day basis. The economic analysis helps people understand the potential value or future economic benefits of specified investments.

The FEA analysis of the SCAI process will be used to support the development of the business case which presents the benefits of using a megaprogramming development strategy. This analysis will be used to support the transition of the SCAI technologies throughout the SWSC organization.

(5) Review and Approve Program

During this step, the management decision is reviewed by appropriate approval authorities for policy, programming and acquisition. If approved, the action plan is implemented.

(6) Execute Functional Process Improvement (FPI) Program Decisions

This process puts into effect the new business plan established for the organization.

The CIM Business Process Improvement Analysis of the SWSC Software Maintenance Process and the Activity Based Costing Foundation Workshop for the November version of the high level SCAI Process Architecture have been accomplished. The workshop evaluated the SCAI approach to software development and maintenance, and appropriate alternatives to that approach from a functional and economic perspective. The following alternatives were developed by the workshop:

(1) *Alternative A* - Implement the SCAI paradigm across the SWSC organization for Space and Warning Control Systems.

(2) *Alternative B* - Using the current baseline process and augmenting it with a compilation of amicable processes and management chances to improve the existing process.

(3) *Alternative C* - The status quo. Continue to do business without making any change in the process.

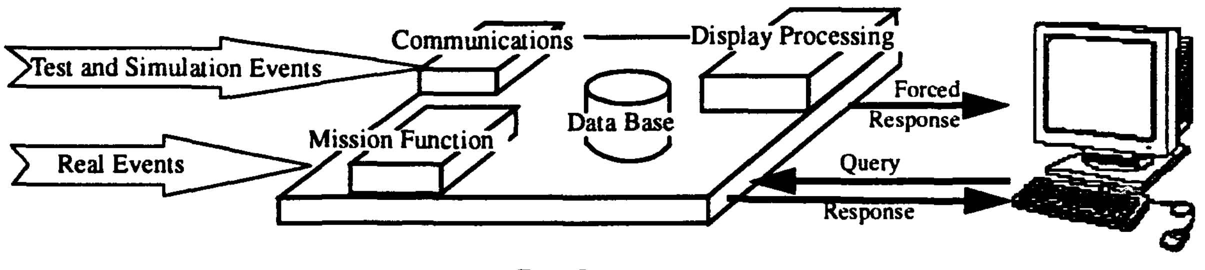
An FEA workshop will be done to accomplish complete analysis of the alternatives.

Design/IDEF, developed by Meta Software, is an automated support technology for developing IDEF models. It is being used on the SCAI to document the high level Process Architecture.

Reusable Integrated Command Center (RICC).

RICC is, a technology developed for the SWSC by TRW under the Air Force Embedded Computer Resources Support Improvement Program, and includes tools for generating Ada code, definition files for applications and some reusable domain components. It was developed in response to the observation that the relatively diverse set of C^2 systems, have a large subset of common requirements. The commonality or requirements suggested that significant system development and maintenance leverage could be obtained by developing reusable software to satisfy those common requirements.

Figure 11 depicts the general functions which are performed by most C^2 centers. In general C^2 centers receive mission status and data messages from external sensor sites and/or other C^2 centers. C^2 centers also output mission assessment, status, and command messages to the external sensor sites and C^2 centers. The Communication function must provide the capability to process the protocols used to communicate this message traffic. There are a number of standard protocols and consequently this function is a good candidate for incorporating into a C^2 infrastructure.



Data Base

Figure 1. Typical Command Center Functionality

The Communication function provides the input and output message processing. The input message function receives inbound external messages from external entities, extracts the various fields from the message, converts the fields to internal data types as required, and validates the data for each of the fields to ensure that the message is valid. Following validation of the input message, the converted and validated data is given to the mission algorithm functions for processing or stored directly in the database. On the output side, the output message processing starts with the receipt of mission data from either the Mission Algorithms or from the Command Center User to be sent to external entities. Then Communication function formats the data into the correct external message format and transmits the messages. Both external input and output message formats are typically unique to the missions being performed. However, the functions performed in input and output message processing are basically common from one C^2 system to another. Thus it is possible to develop generic data driven message processing software which is tailored for specific missions via mission specific database data describing the formats of the messages.

The Mission functions receive mission data from both external message sources and from the operator. These algorithms will process the input mission data and then make the processed data available for output to either external entities via the Communications function or to the operator via the Displays function. Mission algorithms are typically very unique to the specific mission of the C^2 center and consequently are not good candidates for a C^2 infrastructure. However, it should be noted that there are some portions of mission algorithms which are good candidates for the reuse or infrastructure. An example of this is the orbit predication function which is required by many Space and Missile related C^2 centers.

The Data Base provides the repository for information received from external sources and other information generated by the Mission functions.

The final function common to all C^2 centers is the Operator Interface. This function receives mission data from the Mission Algorithm functions, converts and formats the data as required and then outputs the data to the C^2 center operators' consoles. The data displayed occurs as a result of information forced on the user, such as alarms, or data requested by the user. This function also accepts inputs from the operator to initiate, control, and terminate mission processing. The specific displays and control features provided are normally very specific to the particular C^2 mission. However, these displays and controls can be composed from a more basic set of interface objects which are common among various C^2 centers such as menus, commands, forms, graphs, maps, etc. Thus it is possible to develop generic, data driven operator interface software which is tailored for specific missions via mission specific database data describing the interface objects of the displays.

Significant commonality exists between C^2 systems and the common portions tend to be the most technically challenging, costly, and risky areas of the system design and implementation. By developing a reusable software infrastructure which addresses these common requirements, dramatic savings in development cost and schedule time for a given C^2 can be attained.

The C^2 Architectural Goal is to provide a reusable C^2 software infrastructure. Figure 12 is an abstract representation of the layers that are essential to the implementation of a C^2 Center, and the Command and Control Architectural Infrastructure layer represents the functionality provided by RICC and Universal Network Architecture Services (UNAS).

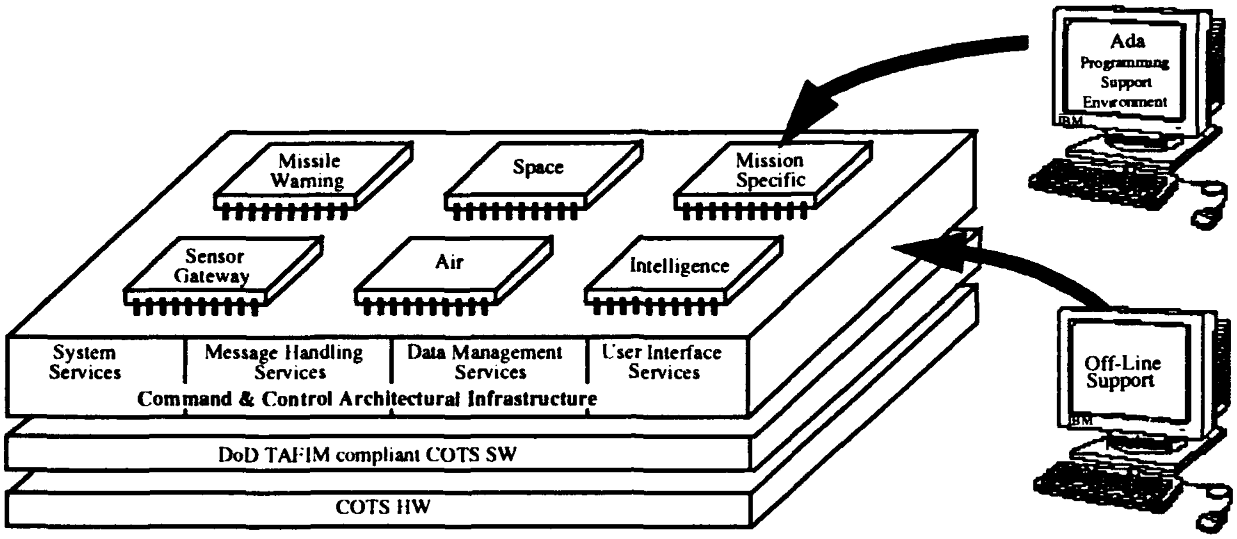


Figure 2. C^2 Architectural Goal

Mission Operation/Information Analysis (MOIA)

Mission Operation/Information Analysis (MOIA) is an analysis method, using $IDEF_0$ notation which looks at the low-level operations of a command center from the point of view of the user. It was created to bridge the classic gap between users and developers; the transition from operational requirements to system requirements. MOIA employs a four phase methodology to analyze and define the activities and information employed by a work center in fulfilling its assigned responsibilities. Each phase is described as follows:

(1) Phase I identifies and describes the activities which are conducted in a work center. MOIA methodology is system independent and captures all activities, manual and automated. The end products of Phase I are:

- a series of MOIA diagrams using $IDEF_0$ diagraming techniques, and

- activity reports.

(2) Phase II associates external inputs with outputs, and vice versa, for each activity at its lowest level. Once the associative process is complete, then candidates for automation or improved automation are identified; allocated to existing or planned systems; and based upon the operations benefit derived from automation and frequency of occurrence, assigned a priority for Implementation.

(3) Phase III involves the analysis of existing program or cost data to determine which candidates for automation associated with a particular system are in the baseline for system development projects already underway. Candidates which are not in a baseline are identified as gaps or shortfalls.

(4) Phase IV develops a plan of action for acquisition adjustments, Pre-Planned Productivity Improvements, or other programmatic actions to satisfy all requirements. In the case of existing programs, this implementation plan recommends appropriate acquisition adjustments if any gap or shortfall was prioritized high enough to warrant changing the program baseline. For new programs, the implementation plan identifies how many of the prioritized items could be paid for with available dollars.

Each of the phases builds upon the previous phase. Once all four phases are completed on a work center, a comprehensive information flow, shortfalls, and implementation plan is available for that center.

Booch Object Oriented Analysis

Like all good object oriented methods, the Booch Method is primarily a means of developing and communicating the design for a system or a family of systems. The development of models of a system occurs in stages that allow for concentration on certain aspects of a system at a time. This approach acknowledges the distinct probability that during the first look at requirements and mapping them to design, requirements understanding will change. Therefore there is a need to continually integrate discoveries into one underlying model, that is, the process iterates between analysis and design.

The Booch Method is accomplished in three steps:

(1) Requirements Analysis provides the basic charter for the systems functions.

(2) Domain Analysis provides the key logical structure of the system.

(3) Design provides the key physical structure of the system and maps the logical structure to it.

Both analysis and design processes are iterative and incremental. When ambiguities or omissions are discovered in the analysis model it is appropriate to return to analysis activity and then continue the iterative process.

TRW Ada Process Model

Walker Royce's TRW Ada Process Model was designed to address design breakage due to early unknowns in large complex systems developments. The key strategies inherent in this approach are directly aimed at the three main contributors to software diseconomy of scale: minimizing the overhead and inaccuracy of interpersonal communications, eliminating rework, and converging requirements stability early in the development life-cycle. These objectives are achieved by:

(1) Requiring continuous and early convergence of individual solutions using Ada as the life-cycle language.

(2) Evolving solution as rapidly as practical to eliminate ambiguities and unknowns in the problem statement by prioritizing development of real code increments of capability.

The objectives are accomplished through the following practices:

(1) Design Integration - Unlike conventional software development which enforces the concept of postponement of all coding until after Critical Design Review (CDR), the Ada Process Model requires the early development of a Software Architecture Skeleton (SAS) which corresponds directly to the top-

level components and their interfaces. The SAS provides a vehicle for early interface definition by compiling these top-level components, and providing adequate drivers/stubs so that they can be executed to evaluate design quality. This forces early baselining of the software interfaces which in turn permits smooth development and avoidance of downstream breakage.

(2) Demonstration-Based Design Review - Again, unlike conventional approaches which rely on paper demonstrations of design correctness, the Ada Process Model employs demonstrations during design reviews to provide validation of design correctness.

(3) Total Quality Management -The use of Ada throughout the life-cycle permits consistent software metrics across the software development work force and the demonstrations serve to provide software developers with tangible progress indicators.

(4) Incremental Development - This well-known software engineering technique is employed because of the need to adjust build content and schedule as more accurate assessments of all factors can be made.

STARS Technologies

STARS Process-Driven Development

To improve software quality and productivity, STARS felt it was necessary to focus attention on the software process being employed and the means used for delivering them. The success of software development organizations is directly related to the quality of the products they produce. When organizations have no training or consistent approach to repeat their results, they are dependent on the resourcefulness of the people they employ and they are likely to achieve pockets of success rather than success across all levels of the organization.

The concepts and approaches which define the STARS Process Driven Development approach for increasing the productivity, reliability and quality of critical government software systems are designed to help organizations and projects to achieve process capabilities such as those characterized by the higher levels of practice in the Software Engineering Institute's (SEI's) Capability Maturity Model (CMM).[1]

What Process Driven Development means to organizations is that:[2]

(1) Organizational processes are formally or semi-formally defined, are globally known and visible throughout the organization, and are adaptable and tailorable to meet project and product goals.

(2) System and software development is guided by a defined process for all activities from the system inception (or beginning of the organization's involvement) through system deployment, evolution, and eventual retirement.

(3) Environments and tools are integrated to support a defined process.

(4) Defined processes promote collaboration and teamwork by making activities, roles, and dependencies clear.

(5) Process definitions are developed, maintained, evolved, and reused with a level of concern and discipline approaching that applied to software products themselves.

(6) Process definition disciplines are improved through manual and automated measurement and feedback techniques.

(7) Process definitions are installed and guidance is available through documentation or tied to automated tool invocation.

1. Technical Report 85.0170, STARS Process Concepts Summary, December 1992, Hal Hart (TRW), Jerry Doland (Paramax), Dick Drake (IBM), William Ett (IBM), Jim King (Boing), Herb Krasner (Krasner Consulting), Leon J. Osterweil (univ. of California at Irvine), James Over (SEI), Terri Payton (Paramax), Co-ordinated by Learning Resource Center IBM, Gaithersburg, Maryland 20879 pg. 2
2. Technical Report 85.0170, ST' RS Process Concepts Summary, pg 2.

The early efforts of the STARS program were to explore innovative solutions to improve the management and control of the complex and varied activities (software process management) required to develop, field, and support Mission Critical Computer Resource (MCCR) software. STARS aimed to establish capabilities to support tailorable process definition and management which involved the ability to define, monitor, measure, control, and continuously improve the activities that make up the software life-cycle processes.[1] Software process management is a key focus of the STARS process-driven paradigm, and the Loral STARS team has focused attention on providing automated support for process-driven project planning, measurements, and automated process enactment to further an organization's ability to manage and control a project and its processes.

Several technologies have been developed to support the STARS process definition and improvement strategy. Process definition requires the specification of the identified processes, where specification includes a process model and a process definition document (also referred to as a process guide) which describes the relevant set of activities, artifacts, and agents; the relationships within and among those three classes; and the behavior of the entire set of entities and relationships. Given these documents, the process can be enacted by humans.[2] Processes may be defined so that they are manually enactable, where a human follows a written procedure, or automated, where a computer coaches a human on what to do next given their current state. The focus on the SCAI project is to support both manual and automated enactment.

Automated Process Support Tools

The enactment of SCAI processes is supported by a number of automated tools, namely: ProjectCatalyst, Process Weaver and SPMS. CAT Compass and Amadeus support project and process management.

ProjectCatalyst

ProjectCatalyst was developed by Loral STARS teammate Software Engineering Technology, Incorporated (SET) and is a family of integrated software components which:

(1) Supports task dispatching and monitoring.

(2) Guides the engineer in following the defined process.

(3) Provides the necessary files and tools to the engineer performing process tasks.

(4) Manages complex parallel process activities.

(5) Facilitate team communication by automatically maintaining task status and tracking task prerequisites.

(6) Reports task, product and milestone status to SPMS.

(7) Gathers effort, schedule and quality measures for presentation by Amadeus.

ProjectCatalyst provides these functions through the use of Process Weaver.

Process Weaver

Process Weaver was developed by Cap Gemini. It functions as the process enactment component of the ProjectCatalyst tools. It provides the interface to the users and supports the management of dispatched tasks.

Software Process Management System

The Software Process Management System was developed with Loral STARS teammate Cedar Creek Process Engineering and supports process definition through process modeling. SPMS supports an Entry, Task, Validation, Exit (ETVX) process definition paradigm, which is conducive to the enactable process definition focus of Information Organization Templates.

1. STARS Workshop Process Management, Dick Drake, STARS Newsletter, Volume II, Number 1, March 1991 pg 5
2. James W. Armitage, Marc I Kellner, Richard W. Phillips, Software Process Definition Guide, Content of Enactable Software Process Definitions, Guide SEI-93-SR-18, Software Engineering Institute, Carnegie-Mellon University August 1993, pages 2-6

SPMS provides the capabilities to model and continually refine an organization's process definition. It also provides the capabilities to instantiate the defined process for a particular project through the inclusion of time and resource scheduling. SPMS interfaces with a project planner package, CAT Compass to provide project management with process-driven project management capabilities. SPMS includes support for:

(1) Process and measurement definition

(2) Process driven plan generation

(3) Scheduling

(4) Resource allocation

(5) Plan simulation

(6) Plan monitoring

SPMS will allow the SCAI project to focus on the vital process issues of cost, schedule, and quality.

CAT Compass

Cat Compass provides the project planning and reporting capability.

Amadeus

Amadeus was developed by Amadeus Software Research and is a software product that automates measurement activities within a software engineering environment. Amadeus consists of several software components, the heart of which is an interpreter. This component monitors significant events such as tool invocations, changes in source files or documents, creation of software problem reports, expiration of time intervals, or changes in collected measurements, and invokes other components called agents, to respond to the events. Some of the agents perform collection of measurement data, while others analyze the data or produce visualizations of the data, or feed it back into the development process.

Cleanroom Software Engineering Process

The Cleanroom Software Engineering approach to quality software development is based on over 25 years of development, verification and testing experience by Dr. Harlan D. Mills and his associates, first at IBM and subsequently at SET. Cleanroom provides a tightly integrated approach from specification preparation through software development to software certification, which brings engineering rigor and intellectual control to the software development process. Cleanroom processes and practices have demonstrated improved programmer productivity and software correctness and reliability.

The fundamental notion of Cleanroom is the necessity for the engineering staff to maintain intellectual control over the project. Intellectual control is the ability to clearly understand and describe the problem at hand at the desired level of abstraction. Cleanroom uses a number of organization and technological strategies to help the engineer maintain intellectual control over the software project.

With Cleanroom, there are three teams that each handle different aspects of the software development process. The teams are as follows:

(1) The Specification Team prepares and maintains the specifications.

(2) The Development Team designs and builds one or more software increments. The resulting source code, prior to any compilation, is turned over to the Certification Team. They are also responsible for isolating and making any changes necessary to the increment as a result of problems. Large projects have multiple Development Teams.

(3) Each Certification Team prepares test cases for an increment and when the increment is submitted for certification they perform the tests and prepare the certification report. The Certification Team executes the code, but does not modify it in any way.

Cleanroom specifications are developed in a spiral manner, with each iteration making the specifications more complete. A Cleanroom Specification is comprised of six volumes:

(1) Mission: A precise statement of the requirements for the software system, primarily functionality and performance.

(2) User's Reference Manual: A definition of the stimuli and responses invented so the software can fulfill its mission. This volume addresses 'look and feel' issues, as well as performance.

(3) Software Function: A black box function that defines software responses in terms of stimuli histories (i.e. in an implementation-free manner). This volume clearly describes the functional behavior of the software.

(4) Specification Verification: A rigorous argument that the software as defined will meet its defined mission.

(5) Software Usage Profile: A Markov Model stating the probability of moving from each usage state to all other usage states. This volume describes how the software is to be used, in terms of sequences of state transitions, according to a projection of how the software will actually be used. Every possible state that a system can be in described, with probabilities associated with all possible next states that can be reached from a particular state as a result of a single transition. The Usage Profile can be represented in terms of a state transition diagram or a matrix. This volume may also include test fragments for each transition, which can be concatenated, as a result of statistically generating a path through the software, into a test case.

(6) Construction Plan: A plan for building the software in a series of increments such that each accumulation of increments is executable by user stimuli. The development of the construction plan requires significant effort to determine a definable series of user executed increments. Increments of 5000 to 10000 lines of code, which require approximately 6 to 8 weeks of effort for a Development Team, typically have been utilized.

Domain Analysis Process Model

Reuben Prieto-Diaz created a systematic, repeatable process for doing Domain Analysis, which can be described as: "systems analysis for a class of systems," or "the activity of identifying the objects and operations of a class of similar systems in particular problem domain". The objective of Domain Analysis is to discover and define domain models and architectures common to a family of applications for supporting pre-planned reuse.

Following is a sketch of the Domain Analysis Process at a very high level:

(1) Select the Domain with the Highest Reuse Potential

- Look at Current Projects: Scope and Define the Domain
- Evaluate Current/Future Needs, Current Practice, Feasibility
- Define Purpose

(2) Top-Down Analysis

- Identify High Level Architecture and Functional Model
- Select Functional Components with High Reuse Potential
- Re-define Architecture (with Reuse in Mind)

(3) Bottom-up Analysis

- Vocabulary Analysis

- Classification Model
- Functional Clustering

(4) Derive Generic Architecture

- Map Bottom-up Functions into Architecture
- Adapt Architecture
- Derive Other Models

Outputs of this process are:

(1) Domain Definition

(2) High Level Domain Model/Architecture

(3) Faceted Classification

(4) Domain Vocabulary

(5) Reusable Structures

The unique aspect of the process is the bottom-up part of the analysis. The contention is that by examining systems and systems documentation, common terms can be derived that are general to the whole domain. More specific terms can be grouped under more general terms called facets. The more specific terms are deemed facet-terms. So, potentially, a domain analyst can uncover generality simply by understanding language. Relationships can be generated between the facets so that "standard descriptors" can be generated for the domain. Standard descriptors can represent generic functions in the domain. For example, an early attempt at reuse was when Booch defined a set of general Ada Components managing standard data types like stacks and queues. A standard descriptor for Booch Components, where parenthesized items are facets, would be: Component of type (Role) consists of or operates on (Structure), has (Concurrency) and (Space_form), performs (functions(s)) on (object(s)), using (Method). The facets for the Booch classification scheme are in parenthesis.

SCAI/STARS/SEI Process-Driven Technologies

Information Organizer Templates

Information Organizer Templates, a mechanism to support the definition of manually enactable processes, has been developed as a result of a Process Definition Technology Partnership between the SCAI project and the Software Engineering Institute (SEI). The partnership was intended to develop prototype products, and collaborate on solutions to technical problems. The templates identify all the information required to make a process manually enactable. In order to avoid overwhelming the potential process definers or process users, the team devised a way to subdivide the information into categories, which also represent convenient stages for the process definer to follow when developing the process definition.[1]

(1) Stage 1: **Process Layout:** Show overall flow of activities and work products defining what is to be accomplished by a given process; available in an enactable process guide for reference as a navigation aid during enactment.

(2) Stage 2: **Process Design:** Include lower level refinements beyond information necessary to just depict process layout, such as the methods that describe how an activity's results are to be accomplished, and the agents responsible for performing the activities. This information is documented in an enactable process guide for reference as needed prior to and during enactment.

1. Linda Parker Gates, Richard W. Phillips, STARS/SEI Technology Transition Experience Report November 30, 1993, Software Engineering Institute, Carnegie Mellon University Pittsburgh, Pennsylvania pg.19

(3) Stage 3: **Process Enactment**: Include the process information that is necessary during enactment to govern the process (such as activity and artifact states) or information that describes specific process discipline actions to perform during enactment (such as when to verify or validate a result, when to record product and process metrics, when to log status, when to communicate status to others, or solicit status from others, and how to determine which activity in the process is to be performed next upon completion of a given activity).

(4) Defining a set of metrics which could verify the successful achievement of the goals,

(5) Establishing an approach for capture and collection of the metrics,

(6) Defining an approach to presenting and analyzing the measurement results,

(7) Instituting a continuous improvement cycle for the measurement process itself.

The approach to metrics definition will be both top down and bottom up. From a top down perspective the project goals will be defined and refined. The purpose of the metrics activity is to determine if the goals are being reached. The goals provide the rationale for the actual measurements collected. You could also think of the goals as providing requirements on the metrics activity.

Appendix C

Megaprogramming: Enabling the Future SWSC Product-Line

The SWSC is responsible for the maintenance of a wide range of space and warning C^2 applications. The SWSC's long-range intent is to manage as many systems as possible within this domain of applications using a coherent process; and the envisioned product-line organization is intended to facilitate this type of management. There are three fundamental prerequisites for a successful product-line organization: a domain manager responsible for all applications in the product-line, sufficient commonality among the applications to allow management of the entire set as a whole, and enabling technology to facilitate the systematic management and engineering work.

Domain Manager

Figure 1. depicts a possible future product-line organization and shows how the three STARS technology areas combine to enable a product-line way of doing business. The SWSC Commander would serve as the Domain Manager in this organization. Ideally, the Domain Manager will be in a position to make trade-offs between individual application objectives and overall product-line evolution objectives. As an example of this type of trade-off, it might be appropriate to build a new reusable asset that will greatly enhance the efficiency of the total organization (enabling the SWSC to build future systems "cheaper, better, faster"), but in order to build this new asset, there might be some impact to the cost and schedule of one or more applications.

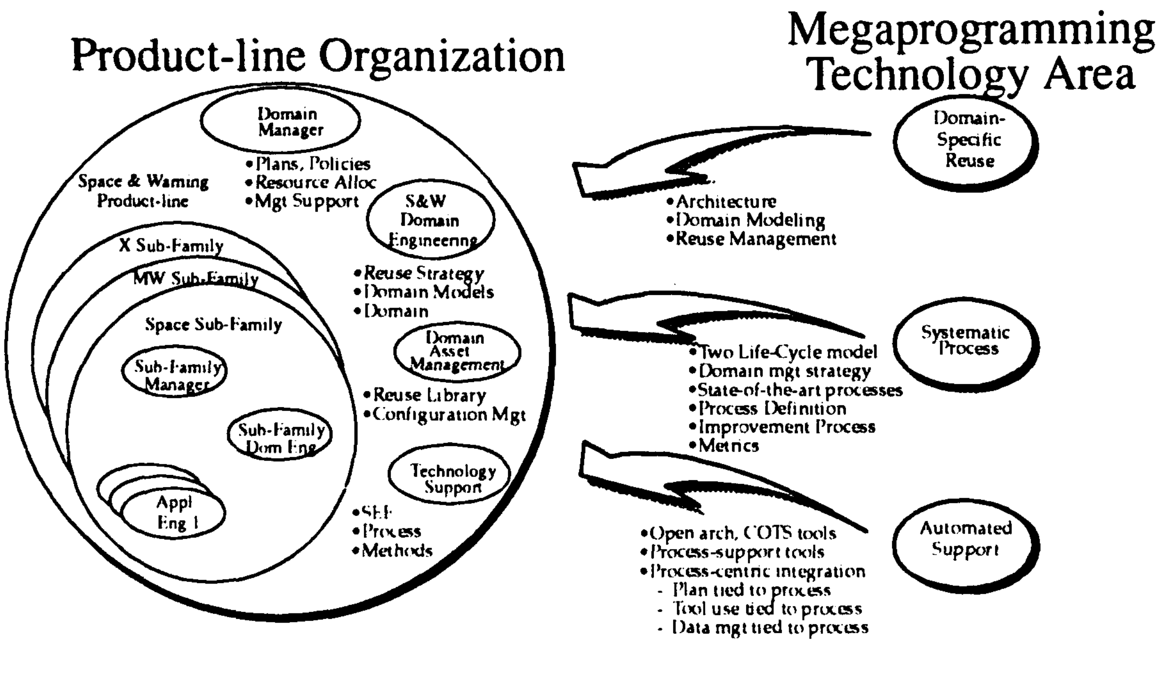


Figure 1. Megaprogramming: Enabling a Future Product-line Organization

Application Sub-families

Applications in the SWSC product-line will be developed and maintained using a common Application Engineering process that maximizes the reuse of common product-line assets, including process components, software components, models, software engineering environments and tools, etc. As shown in Figure 1., these applications may be further divided into sub-families based on the opportunity for additional layers of reuse.

Domain Engineering

Domain Engineering is the engineering arm of the organization charged with determining the commonality in the requirements of the domain's applications and deriving and evolving the best architectural approach to building those applications - which includes identifying reusable software components, code generators, and the like.

Domain Asset Management

Domain Asset Management provides storage and management of all product-line assets of all types (process components, software components, models, specifications, etc.), facilitating the proper use of those assets across the entire product-line.

Technology Support

Technology Support identifies, develops, evolves and supports the organization's common processes and methods. It also provides the software engineering environment, which provides automated assistance for applying the processes.

Megaprogramming Technology Emphases During the Demonstration Project

To prepare for this type of organization, the SWSC is working with STARS to mature its technological approach, using the SCAI Demonstration Project as its primary vehicle. As discussed below, the SWSC and its existing contractor base have already made substantial progress towards megaprogramming; thus, more attention is being given to some areas than others.

(1) **Domain-specific Reuse**

Prior to forming the partnership with STARS, the SWSC had already established an excellent head-start in this technology area through its work with TRW on a reusable architectural infrastructure - culminating in the Reusable Integrated Command Center (RICC). This provided an architectural starting point for the SCAI. The SWSC's primary focus in domain-specific reuse during the SCAI project is systematizing the use of the architectural infrastructure and building up Domain and Application Engineering (DE and AE) processes. Heavy emphasis is being given to various types of modeling as a means of capturing the requirements for the SCAI application, specifying its architecture, and following these models through to the executing system. The abstractions being developed during this activity are being chosen to insure that the models will form the starting point for the product-line's Domain Engineering modeling work.

(2) **Systematic Process**

This technology area is the one that is receiving the most emphasis by the Air Force/STARS partnership. The intent is to build up a coherent process for the SCAI, with emphasis on Application Engineering, and to establish a repeatable product-line method for recording, modeling, managing, and applying the process.

(3) **Automated Support**

STARS is working with the SWSC to build up an integrated software engineering environment (SEE) to support the SCAI process. Notably, STARS is supplying two new process support tools and is supporting the integration of those tools with the rest of the SEE to provide automated support for the process.

Appendix D

Iterative Technology Assimilation and Evolution

Applicability of the STARS CFRP

One of the key principles espoused by STARS is the Plan/Enact/Learn paradigm, as detailed in the STARS Conceptual Framework for Reuse Processes (CFRP), as illustrated in Figure 2.

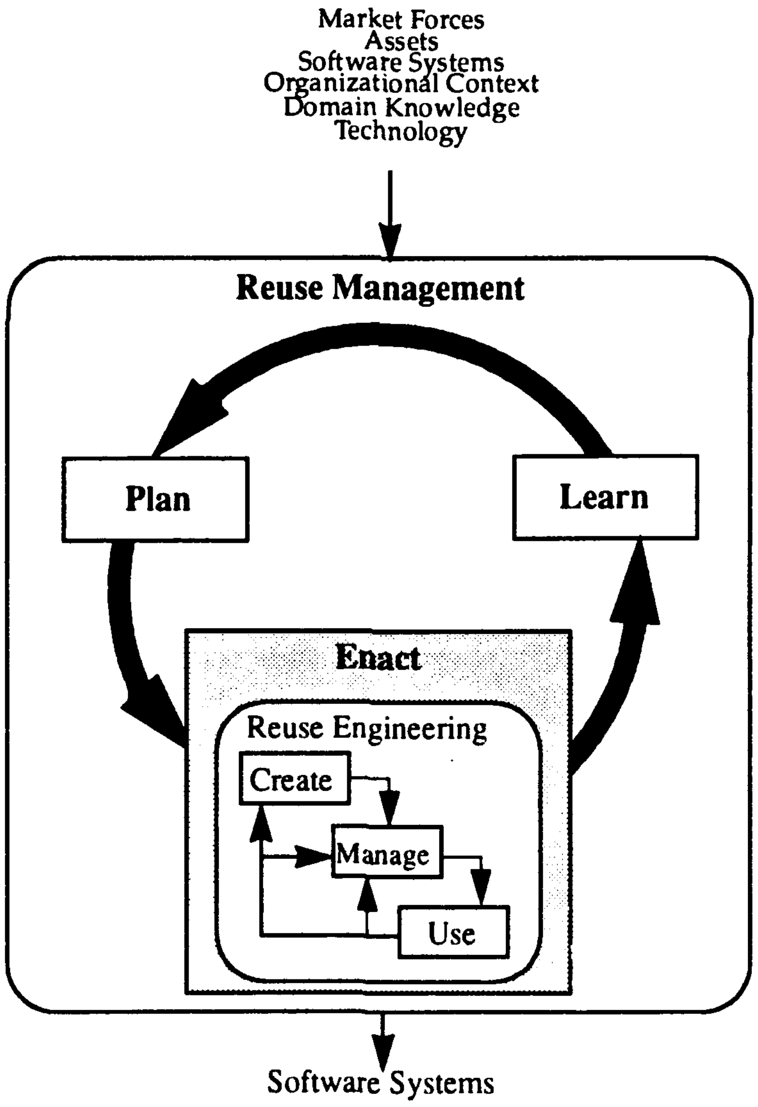


Figure 2. Conceptual Framework for Reuse Processes (CFRP)

The essence of this framework is that introducing anything significantly new is best viewed as an iterative undertaking; and that if the intent is to reuse the results, the iteration must be consciously managed. Thus, incremental changes are planned, the changes are applied, and the experience gained in applying the changes is used to adjust the plan for the next iteration.

For the SCAI project, it is meaningful to consider the CFRP from various points of view, notably:

(1) Formulating an overall technical approach,

(2) Formulating specific approaches in each of the four experience areas,

(3) Developing reusable assets for the domain, and

(4) Developing the SCAI application, including reuse of domain assets.

The first two deal with technology transition and process definition, which were the main activities addressed by the team during the Preparation Phase. The last two deal with producing application and domain products. At first glance, the CFRP seems to pertain primarily to items 3 and 4, but the SCAI team's experience has demonstrated that it in fact pertains equally well to items 1 and 2. If this fact had been better appreciated at the outset, the team might have been able to achieve consensus on the approach more quickly.

In general, the team has found that an iterative approach permeates nearly all significant technical activities.

The following paragraphs briefly discuss the Preparation Phase experience from each of the above points of view, in the light of the CFRP:

(1) Formulating an overall technical approach:

Several characteristics were identified by the SMX directorate that were considered essential to the approach. It must:

- allow for early buy-in by the customer,
- allow for iterative development of the system,
- support product-line development, and
- be evolutionarily in nature.

The CFRP iteration began with the Learn point of the cycle with a joint review of the SCAI objectives and a review of the existing SWSC and STARS technologies and progressed into a cycle of synthesizing candidate solutions, discussing and documenting them, assessing their viability, and moving on to the next iteration. Each successive iteration would typically involve refining/adjusting the solutions defined in the prior iteration, and integrating new aspects of the approach not previously considered.

At the outset, with the large amount of learning needed on all sides, it was very hard to develop specific plans for developing the approach. Given the number and sophistication of the technological changes contemplated, a large number of iterations were needed before the approach really began to gel. In fact, the first attempt to capture the total approach in a coherent description was in Version 2.0 of the SCAI Demonstration Project Management Plan (SDPMP), published at the end of the Preparation period (10/15/93). Although SCAI development began shortly thereafter, the project realizes that a significant amount of iteration remains ahead - and in fact, the iteration really continues throughout the life of the product-line.

The following discussion provides more detail about how this iteration has progressed to date.

The multi-organizational project had to understand the variety of SWSC incumbent technologies, the organizations in which those technologies were used, and the STARS technologies, before the participants could begin the process of integrating the technologies into the SCAI technical approach. The approach needed to integrate the STARS technologies with the SCAI incumbent technologies such that the resulting approach met the characteristics identified by the SMX directorate. The steps that the project planned to iterate through to develop the approach were:

- Understand the organizations contributing to the SCAI project.
- Decide/Plan how to integrate incumbent technologies with STARS technologies.
- Test the merged technologies in Pilot Projects
- Record the processes for applying these merged technologies.
- Learn lessons about the merged technologies and record the lessons.
- Incrementally, integrate the tool-set associated with the merged technologies.
- Enact selected merged processes

(2) Formulating specific approaches in each of the four experience areas:

The CFRP applies equally well to the activities the project has followed in developing the specific approaches and instrumention of those approaches in each of the four experience areas discussed in this report. In brief, the project has experienced the following:

- **Domain/Application Engineering**: Defining the DE/AE approach during the Preparation Phase involved acquiring both domain and technology knowledge (learn), defining/refining the candidate approach and assessing its viability (enact), and determining which aspects to address in the next iteration (plan). At each iteration, the team matured its understanding of both the domain and the technology. Further, at each iteration, the team added more depth to the approach. This experience is recounted in detail in the DE/AE section.

  Another aspect in which the CFRP applies is to the resulting DE/AE approach itself. Domain Engineering for the SWSC is viewed as an iterative activity, starting with the Application Engineering for the SCAI and using the SCAI AE products as the start of the DE iteration. As new applications are built, the new experience gained is fed back into the DE products.

- **Process Support**: The project has iterated on its method for defining process. As the overall project approach matured, several iterations of the process architecture occurred. At each stage, the team learned more about how the $IDEF_0$ process modeling method should be applied and how it should be combined with other modeling methods. At the end of the Preparation Phase, a plan was put in place to use the adopted process modeling approach to build up the SCAI process incrementally during the Performance Phase - starting with the Specification Process. The modeling, and later the enactment, will be done with automated assistance from a STARS-sponsored process tool suite. It is expected that both the process and the tool suite will be revised incrementally as the project gains experience with their use.

- **SEE Support**: SEE assembly and integration is being performed iteratively as well, especially in view of the iterative development of the process.

- **Metrics**: The metrics iteration started with the definition and documentation of the SCAI project goals. As the project proceeded, the goals were iterated to reflect experience. In addition, the metrics instrumentation in the SEE is being built up iteratively.

(3) Developing reusable assets for the domain

The SWSC has already done a great deal of iteration to build up its architectural infrastructure capability. The RICC tool set, which is an outgrowth of prior production work, has gone through two pilot activities to assess its applicability to the SWSC domain. At each stage of the iteration, the experience gained was used to enhance the tool set and the methods used to apply it. Further refinements are anticipated as the SCAI Application Engineering proceeds through its planned incremental development cycle.

(4) Developing the SCAI application, including reuse of domain assets.

During the Preparation Phase, the team progressed through several iterations of its domain and application modeling work. At each stage, the team enhanced its domain knowledge and increased the depth of the models as well. At the close of the Preparation Phase, the team had begun its specification work, using these models as a basis. During the Performance Phase, three SCAI increments are planned. At each stage, experience gained will be used to update and refine both the models and the specifications.

Appendix E

Preliminary DE/AE Approach

This Appendix provides additional details on the SCAI team's DE/AE experience during the Preparation Phase. This information will provide insight into the problems faced by the SCAI team while they worked to develop their approach, as well as the preliminary solutions to some of the problems. The material should be of interest to practitioners in this technology area.

It is important to recognize the preliminary nature of this information. At the conclusion of the Preparation Phase, the team had established an overall DE/AE approach, had defined a high-level project process architecture, and had initiated an iterative cycle of formal process definition. Accordingly, much of this material is only partially developed. During the next experience interval, the DE/AE process definition will continue to mature - with the help of early enactment of selected process components on releases 1 and 2 of the SCAI application. The next version of this Experience Report will update this material.

This appendix includes the following major sections:

(1) Selected DE/AE Approach Topics

(2) Creating a DE/AE Process

(3) Preliminary SCAI DE/AE Process

Selected DE/AE Approach Topics

This subsection discusses several topics that have proven especially important to the team in guiding the approach definition work during the Preparation Phase.

Architectural Infrastructure Chip Model

Figure 1 illustrates a typical Command and Control (C^2) system of the type maintained by the SWSC.

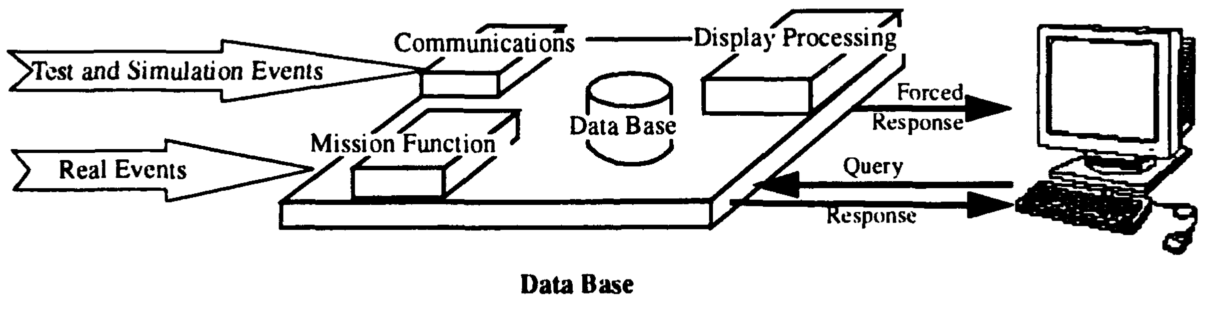


Figure 1. Typical Command Center Functionality

The SWSC has been attempting to determine a common architectural strategy for such systems for several years, culminating in the RICC architectural infrastructure approach, which is detailed in Appendix B.

The so-called "Chip Diagram", shown as Figure 2 represents the current architecture concept. The Demonstration Project is basing the SCAI application on this infrastructure, which is enabled by the RICC tools.

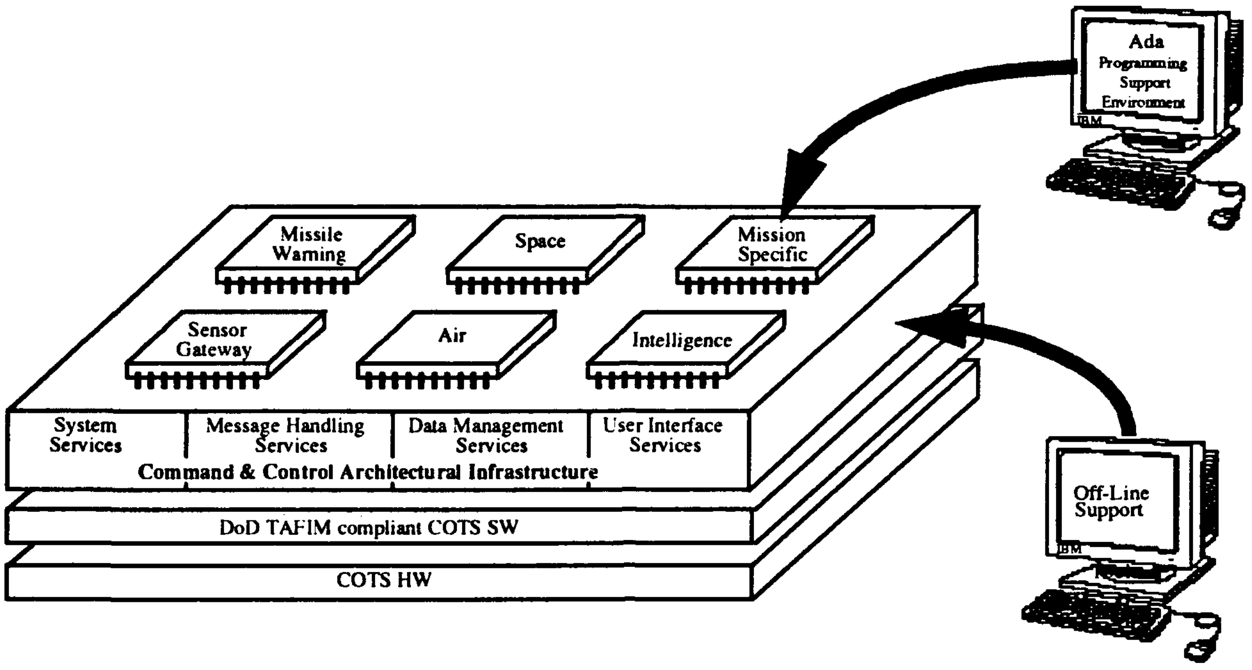


Figure 2. C^2 Architectural Goal

Layered Model Framework

The RICC two layer C^2AI chip model, shown above, was abstracted into the layering scheme, which is called the Layered Model Framework, by adding extra model layers to identify types of reusable components other than the common service routines, and to further insulate the domain components from change.

The RICC Infrastructure Services, identified as Common Services in Figure 3, are services which are applicable to the whole C^2AI domain. As new systems are analyzed and reengineered via the Domain Engineering /Application Engineering (DE/AE) process, the intention is to identify multiple instances of existing application services, and generalize them into new Common Services. Common Services act as Servers to Requesters in layers above the Common Services Layer.

An Algorithmic Layer, containing common mathematical services, potentially applicable to all SWSC systems, has been defined. These Algorithms, such as for Orbit Determination, are applicable to more than one system in the SWSC domain, but probably not applicable to the entire C^2AI domain.

Both the Common Services and Algorithmic Layers act as Servers to the Mission (or Event Layer). The Mission layer models events the system must respond to. It is presumed that there are few missions that are general to all systems in the domain. Therefore, most Missions reside in the Mission Layer of an Application Architecture Model, not the Domain Architecture Model.

A Domain Requirements Model (DRM) is a logical, processor-independent, representation of the common objects in all systems in the domain. Common domain requirements are maintained in the Cleanroom Six Volume System Specification for the domain.

If the layering structure and rules for using the layers in the domain are consistent across that domain, then the DRM can be built iteratively. The characteristic layering structure chosen for our Domain Models should be applicable to the whole SWSC domain and the Space Domain Model Common Services Layer should remain relatively free from change, even as the scope of the Domain Analysis is extended beyond the Space Subdomain.

The anticipated value of this approach is threefold:

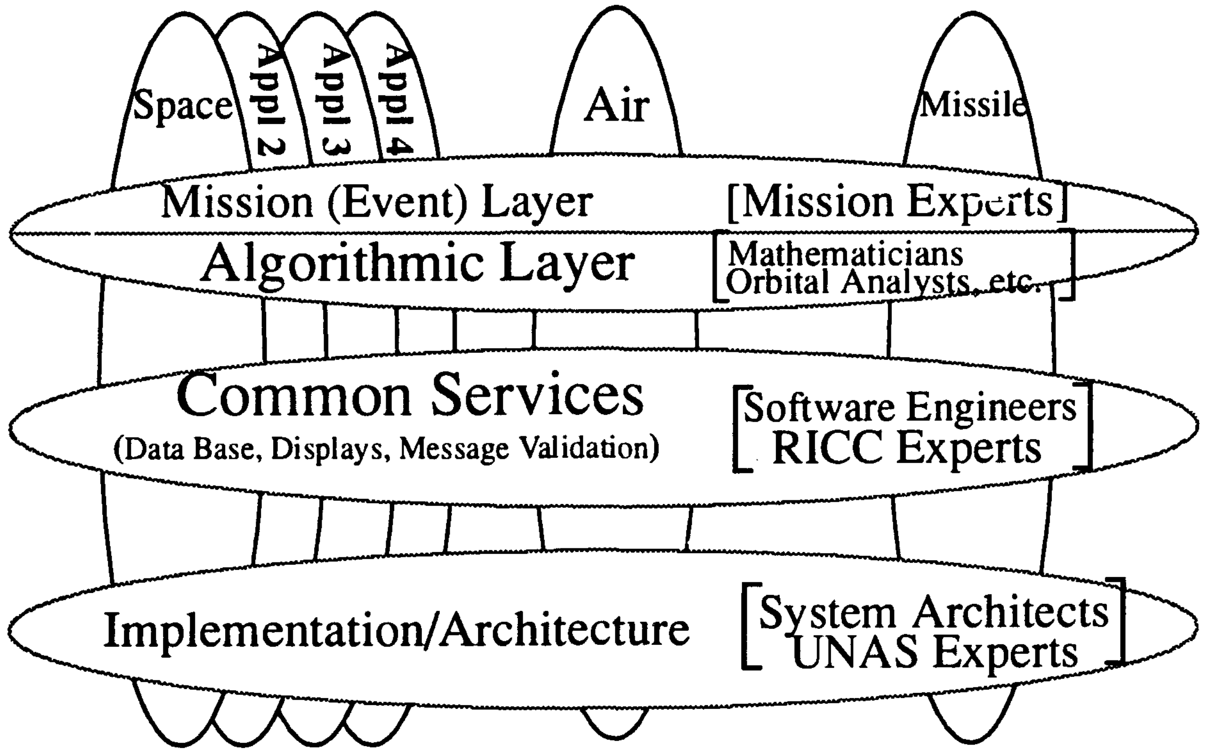

Space
Appl 2
Appl 3
Appl 4
Air
Missile
Mission (Event) Layer
[Mission Experts]
Algorithmic Layer
[Mathematicians Orbital Analysts, etc.]
Common Services
(Data Base, Displays, Message Validation)
[Software Engineers RICC Experts]
Implementation/Architecture
[System Architects UNAS Experts]

Figure 3. Layered Model Framework

(1) The relatively cheap a-priori identification of dominant domain characteristic is substituted for the expensive a-priori Domain Engineering and asset creation for the entire domain.

(2) Domain Models can be validated by use on single systems. Errors are caught before they become expensive. This is consistent with modern Software Engineering Principles.

(3) Other organizations can examine SCAI processes and technologies after they are validated, and accurately predict the cost of making the same technology transition, without factoring in the cost of a wide-scoped Domain Engineering.

The DRM layering scheme not only helps identify reuse opportunities, but also identifies the type expertise that can be localized in product-line functional organizations.

Experts who understand the Missions in the SWSC domain need only understand the Mission Layer of the Domain Requirements Model, and need not be software engineers or system architects. These mission domain experts can collaborate to identify and abstract common missions across systems; they would benefit from having worked on the teams that did the CIM modeling and MOIA predecessor work.

Experts who understand orbits, trajectories and the mathematics behind these topics, can be shared across appropriate SWSC supported missions, as part of an organization supporting the Algorithmic Layer.

Experts who understand typical software engineering tasks like relational data bases, or message parsing, can be considered as a pool available to all SWSC missions. They conceptually belong to the Service Layer of the SWSC, and should be responsible for upgrading and extending RICC services, or substituting services equivalent to RICC.

Architects, experts in creating UNAS System Architecture Skeletons and running trade studies to evaluate system performance, can be viewed as a pool of experience across the SWSC. These experts should also understand how to identify common templates that enforce common structure or mapping rules when converting the DRM into the

Domain Architectural Model (DAM). These experts support the implementation/architectural layer of the DRM, which is a conceptual layer only, and does not relate to actual Ada components, as do the other DRM layers.

Also being advocated, was a Domain Architecture Model, constructed using the Shlaer-Mellor Recursive Design Technique. The architecture model structure, though not the modeling process, was selected by the SCAI project. This structure was consistent with the architectural model structure advocated by Sholom Cohen and his Feature Oriented Domain Analysis.

SCAI Relationship Between Domain and Application Models

Domain and Application Requirements Models (DRM and ARM) are viewed as a single Booch Class Model. Referring to Figure 4, classes (and resultant Ada Packages) are either specific to a single application or general to all applications. Differences between applications are captured using child classes. Similarities between systems are captured using parent classes. Object Scenario diagrams in the Event Layer are identified as either common to the whole domain or specific to a single system. For example, the Manual Orbit Determination Scenario is common to Systems A, B, and C. It uses an Observation object which is also common to all three systems. However, Manual Orbit Determination also uses a Sensor object, which has different children for Systems B and C.

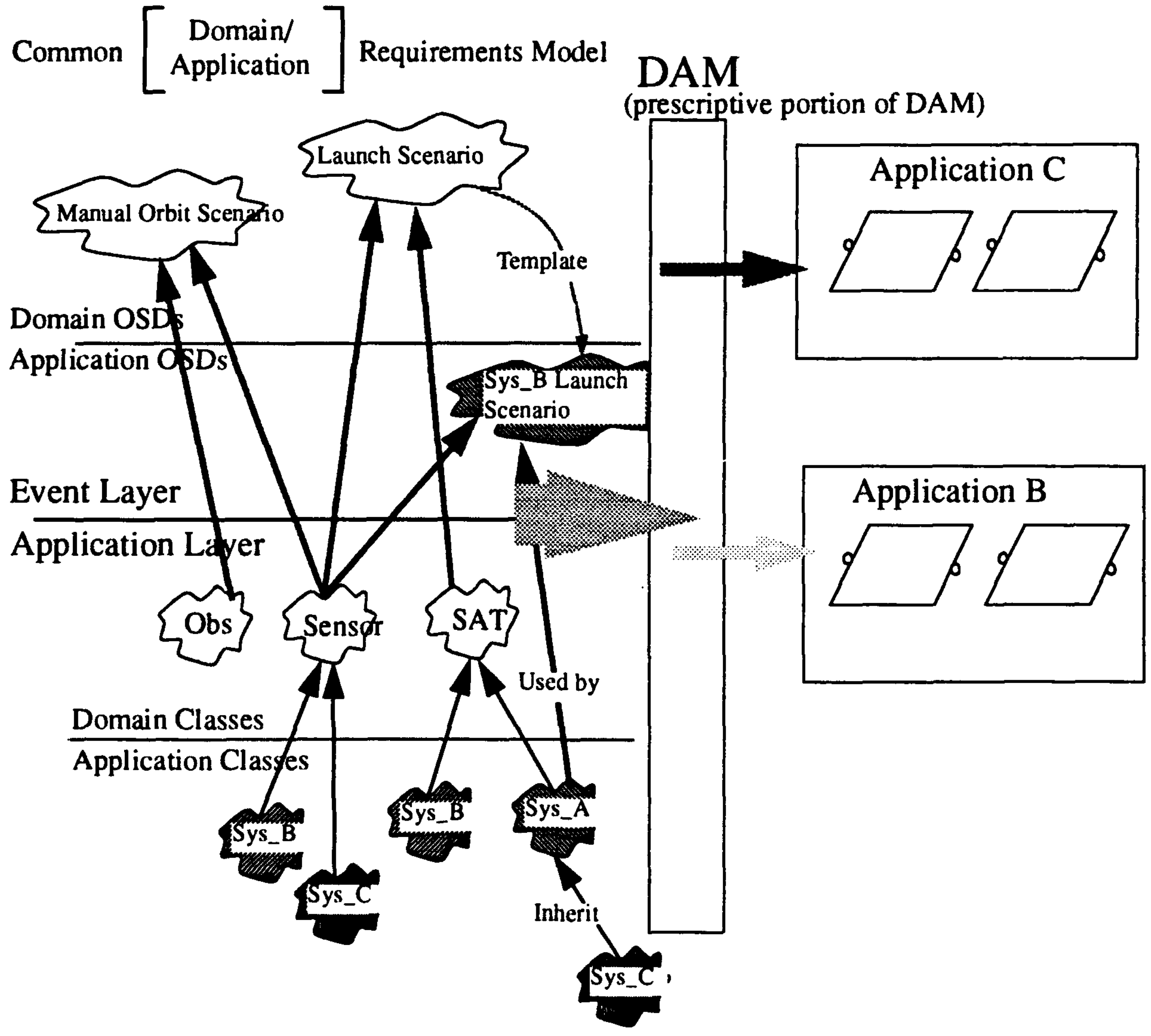


Figure 4. Domain/Application Model Relationship

SCAI Relationship Between Requirements and Architecture Models

Referring again to Figure 4, there is a mapping shown between the requirements model and the application code of two applications in the domain. This mapping is accomplished in accordance with the "prescriptive" portion of the Domain Architecture Model (DAM). This portion provides rules for transforming the requirements model into application software. Theoretically, if the requirements model is complete enough, and if the architectural mapping rules are rigorous enough, this mapping could be automated - i.e., the application could be built from the requirements model alone. While this is not a current focus of the SCAI team, interested readers are referred to the Shlaer-Mellor Recursive Design methodology, which claims to enable such automation.

Not shown explicitly on this diagram is the "descriptive" aspect of the DAM, which is intended to describe the system to human analysts attempting to gain intellectual control over how applications are constructed.

The SCAI team views the descriptive portion of the DAM to overlap heavily with the DRM. As can be seen from Figure 4, some aspects of the architecture of the applications is visible from the OO model. Ideally, one will be able to easily trace between the as-built application software and the requirements model which led to it.

Creating a DE/AE Process

Based on the experience in iterating on the DE/AE approach, the team has established a working model of how to go about defining the DE/AE process (i.e., a formal, well-defined process embodiment of the approach). It should be pointed out that this amounts to a hypothesis, since the team was still at the early stages of formal process definition at the conclusion of the Preparation Phase. It i nonetheless offered as a product of the teams' experience to date.

The following is an outline of the approach to defining the DE/AE Process (Iterate through the following steps):

(1) Develop/Refine and Document the High-Level Approach - including the overall project process architecture (showing how the DE/AE process works in context with the rest of the project process).

(2) Define/Refine the Process Activity Flow: this describes the work flow of the major process activities, including the identification of the key artifacts being communicated among the activities.

(3) Define/Refine the Engineering Artifacts: this describes each of the key artifacts mentioned in the Activity Flow (such as the Cleanroom 6-vol Specification, Object Model, and Architecture Model) within the context of an overall framework; at this point, the layering notions associated with the RICC architectural infrastructure are introduced, since they pervade all the artifacts.

(4) Define/Refine the Mapping between the Artifacts: this specifies how the artifacts tie together into a cohesive whole.

(5) Select/Adjust an Architectural Framework.

(6) Detail /Improve the Process using Information Organizer Templates: this captures the process to the level needed for practitioners who are using it to build product-line applications. Rather than attempt to define the entire process at once, an incremental approach is followed. Modest portions are chosen for detailing, and the experience gained in executing the selected portions is used to guide further definition activities.

(7) Use the Process and Provide Feedback to the process definition activity to help adjust the approach and improve the process.

Preliminary DE/AE Process

The prior section presented an iterative high-level "process for defining an organization's DE/AE process". This section traces this meta-process for the SCAI DE/AE process, illustrating each step with considerations and examples taken from SCAI experience.

This is clearly not a complete process yet. Further elaboration will continue throughout the project. One of the main sources of practical input that will help mature these notions is the ongoing SCAI engineering work.

Develop/Refine and Document the High-Level Approach

(1) Two Life-Cycle model - as interpreted for the SCAI

Figure 5, a version of the STARS Two Life-Cycle Process Model shows the Domain Engineering Process occurring in parallel with the Application Engineering process and suggest that existing systems be analyzed, via a Domain Engineering Process, to identify reusable structures prior to the initiation of Application Engineering. Reusable components are made available before the start of systems implementation.

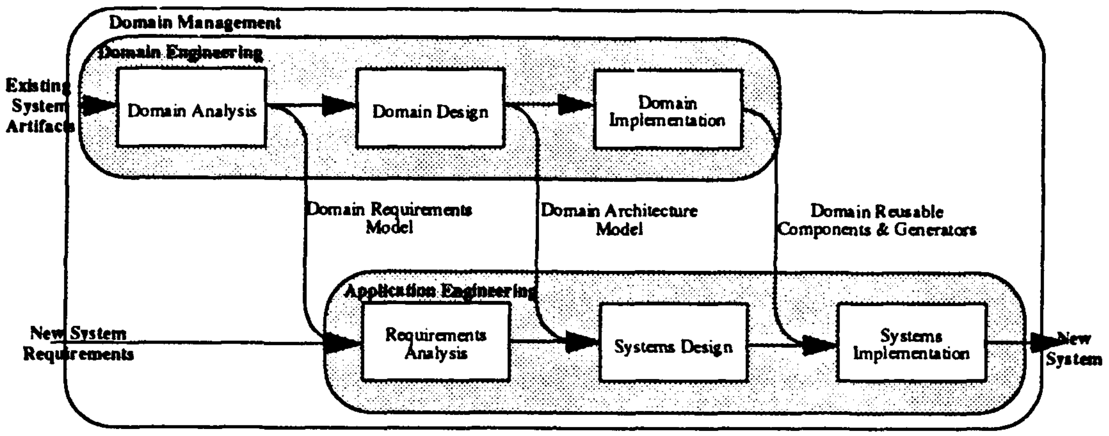


Figure 5. Two Life-Cycle Process Model

For the SWSC domain, the SCAI Team postulated that given the existence of the RICC technologies, and the extensive experience the space and warning domain experts had in analyzing systems in the C^2AI domain, an informal domain analysis for the SCAI C^2 domain had already been done. The MOIA and CIM modeling efforts were also providing space mission operational models as input to the Space domain modeling effort. It was determined therefore that the initial project efforts should emphasize the development of Application Engineering processes, and artifacts.

The approach taken to verify that the RICC C^2 architectural infrastructure could be applied to the space domain was to perform a pilot in which two representative space mission capabilities (threads) were chosen for implementation using the infrastructure. These were Satellite Ground Tracks and Sensor-Satellite Look Angles.

The SCAI approach was to develop a model of the space portion of the C^2 domain because the entire C^2 was too large. The team developing the Space Domain Model acknowledged the RICC as the SCAI architectural infrastructure, and took an incremental approach to developing both models and modeling methodology.

As a result of the informal domain analysis that had been performed prior to the beginning of the SCAI project, the DE team adapted the STARS Two Life-Cycle process to reflect the SCAI approach to building modeling artifacts. (See Figure 6.)

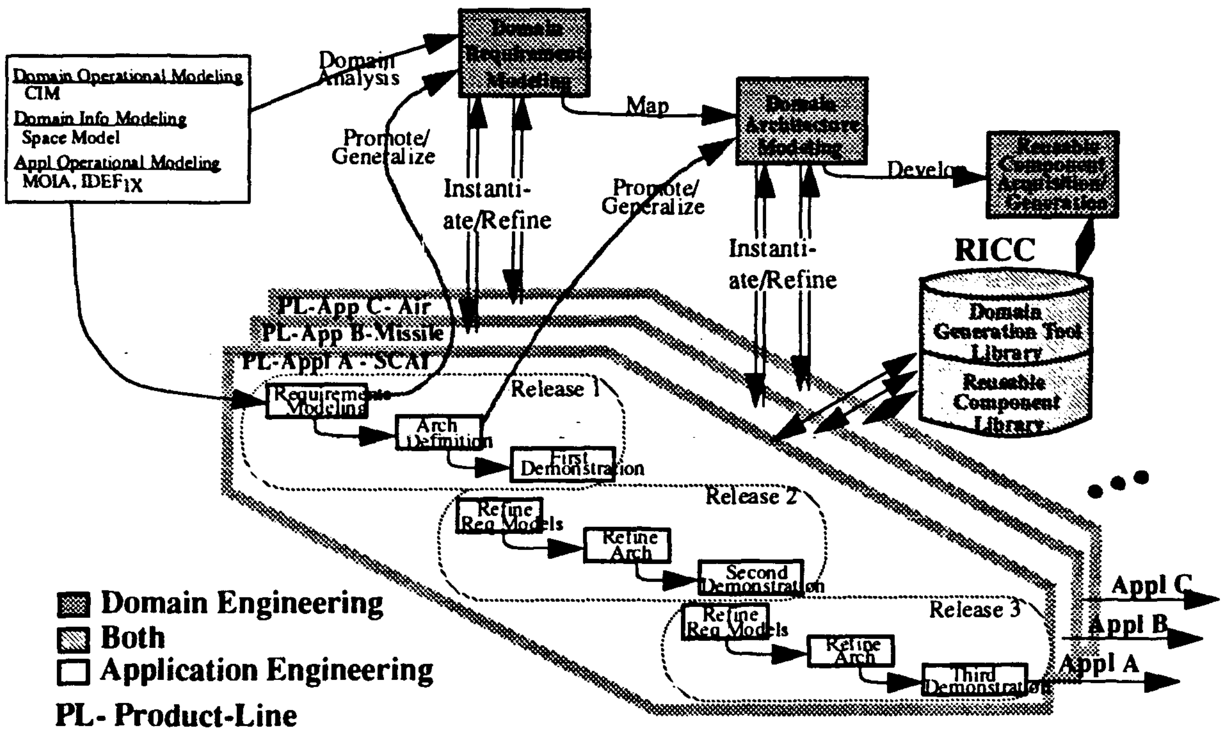


Figure 6. STARS SCAI Two Life-Cycle Process

The SCAI DE Approach consists of iterating through the following steps:

- Define the layering scheme for the Engineering Models.
- Define application-level models in support of Application Engineering. (MOIA and IDEF1x models are detailed models to support the development of the DRM.)
- Promote the application-level models to the domain-engineering level after the first SCAI release and generalize them. (The CIM Space Domain Process Model is used to generalize the DRM because its broader scope identifies users as well as systems).
- Refine the domain-level models through a process of iterative instantiation/refinement during releases 2 and 3. Continue this process of instantiation/refinement for follow-on product-line releases.
- Modify future application models output products to support maintenance and continued development of the Two Life-Cycle models.
- As reusable components are validated by use in a particular Release, they are saved in a Reusable Component Library. Off-line RICC Program Generators are also viewed as reusable assets.

No claims can be made that this iterative approach to Domain Engineering will work for other domains that are less well understood, or not subject to layering.

(2) High-level progression of engineering artifacts.

As part of capturing the high-level approach, the team developed the general artifact progression summarized in Figure 7. The definitions of the Cleanroom specification volumes can be found in Appendix B. The DE/AE operations models, process models and information models form the foundation for the System level Cleanroom specifications. OO analysis yields the high-level object oriented model which identifies the major classes and objects. Based on project management objectives and based on the coherence of clusters of objects, the OO model is divided into release groupings, corresponding to major incremental releases of the application. For each release, the system specification is refined into a release specification, again using the Cleanroom specification techniques. The release specifications, in turn, launch the release development activities.

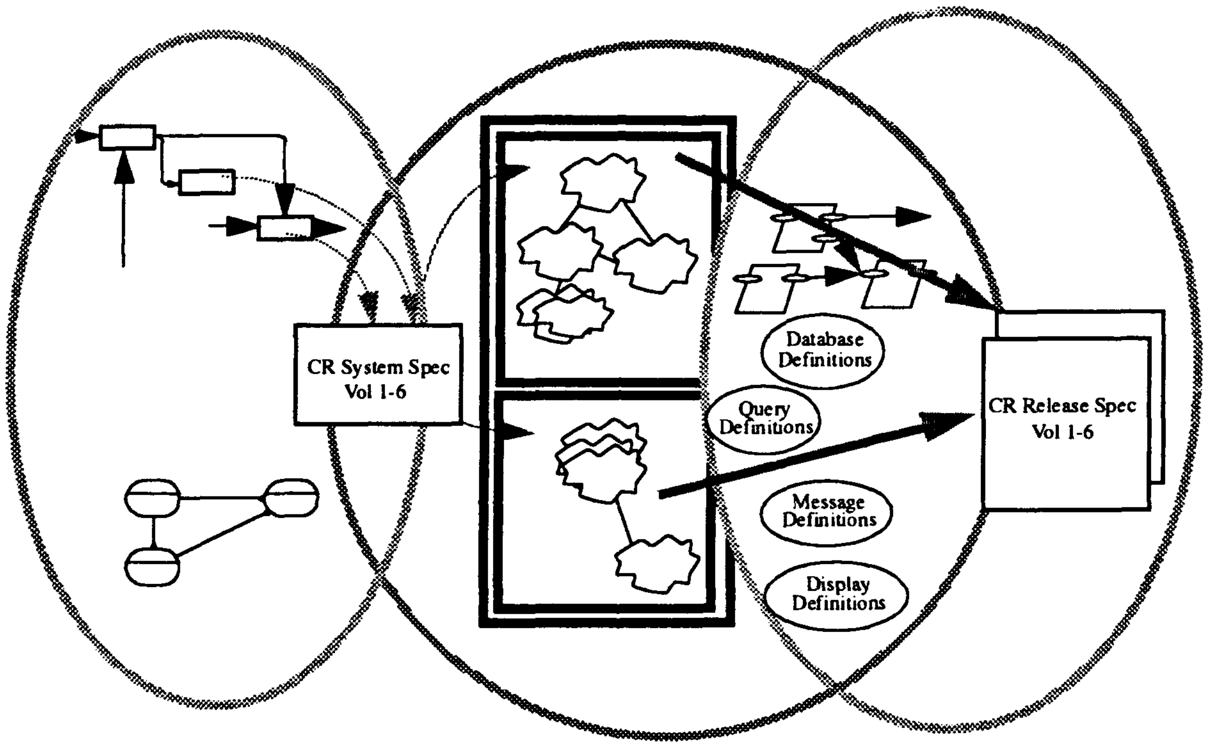


Figure 7. Engineering Artifact Progression

Define/Refine the Process Activity Flow:

Coupling the features of the above three methods/technologies (Cleanroom Software Specification, Booch Object Oriented Analysis, Ada Process Model) resulted in the activities shown in Figure 8 which provides a high level description of the integrated process. Each activity is briefly described below.

(1) Gather System Requirements: Collect information from various sources and scope the system requirements. Resolve requirements issues and clarifications iteratively.

(2) Prepare/Refine Specification: Define the domain/system level requirements. Once the domain/system level requirements are captured, create a Release-level Specification. The focus is on producing a Mission Specification (Vol 1) and a Usage Specification (Vol 2). This activity is coupled with the "Develop/Refine Object Model" activity.

This activity is re-entered (the Refine part) based upon:

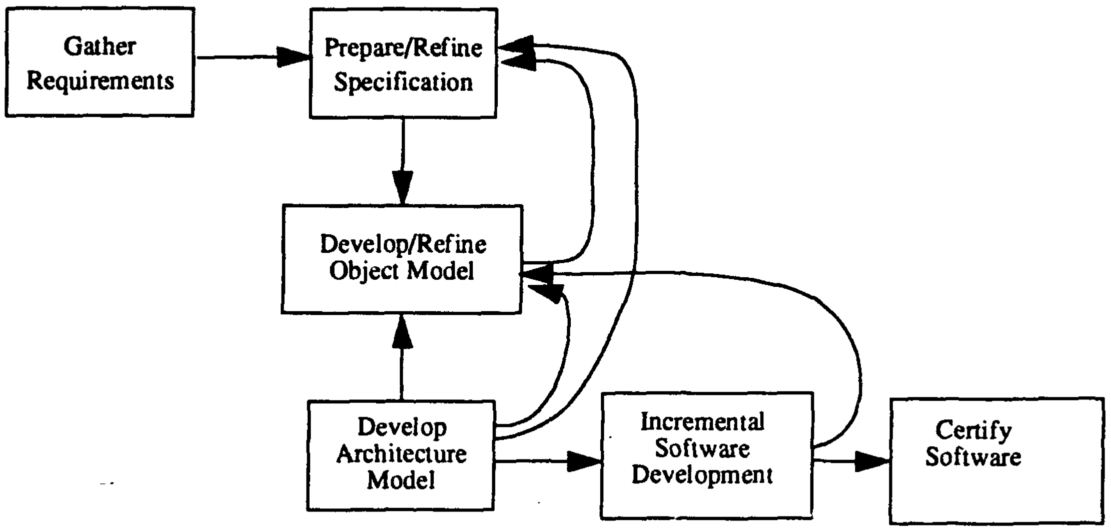

Gather Requirements
Prepare/Refine Specification
Develop/Refine Object Model
Develop Architecture Model
Incremental Software Development
Certify Software

Figure 8. Engineering Workflow

- The Logical View architecture definition. The Cleanroom Black Box Spec (Vol 3) is written based upon this input. The Black Box Specification is validated against the Class Model Object Scenario Diagrams.
- User feedback related to requirements as driven by demonstrations.

Once the Object Model & the Architecture Model are baselined, the Cleanroom Black Box Validation, Usage Profile & Construction Plan (Vol 4-6) are then created.

(3) Develop/Refine Object Model (Logical View of the Architecture): This activity is a Domain Engineering activity. It is presumed that the SWSC mission is to generalize and reengineer existing systems in the SWSC domain. An unabstracted object oriented model of the existing system is created. The model is then abstracted to create a Booch Class Model, or Application Requirements Model. This abstraction activity can be viewed as "reengineering." Or, if a Requirements Model already exists for a different system in the same domain, then the Domain Requirements Model is extended and reabstracted to incorporate the characteristics identified in the unabstracted new system model. The goal of this abstraction activity is to identify the key abstractions (classes) and key mechanisms (scenarios) of the problem domain. This analysis is validated by creating an initial System Architecture Skeleton (SAS -part of the Architecture Model), and executing it, "pumping" data into key scenarios. Next, the class structure is completely elaborated, and any remaining non-essential scenarios are defined. This step is highly iterative with the "Develop Architecture Model" activity. These two activities serve to support the identification of a candidate Domain Architecture Model. At the conclusion of this step, the Domain Requirements Model:

- can be effectively baselined;
- can be used by "Prepare/Refine Specification" as input in creating the Cleanroom Black Box Specification (Vol 3);
- serves as a starting point for the "Incremental Development" activity which will utilize the RICC tools.

(4) Develop/Refine Domain/Application Architecture Model (aka Physical View of the Architecture): Initially, the key abstractions can be used to create a preliminary Architecture Model. As the Object Model solidifies, candidate architectures can be explored to validate key scenarios. A candidate architecture is chosen and preliminary design decisions are demonstrated in a critical thread SAS demo. This demo shows that Quality Performance Requirements (QPRs) can be met in light of critical scenario operations. The chosen Architecture is then baselined. Application SASs must imbed Domain SAS. Developers of Domain SASs should examine the Trade Studies which are recorded as part of the Domain Architecture Model.

(5) Incremental Software Development: Develop displays, messages, database queries and mission components within the structure of the SAS. This activity is iterative with the Develop/Refine Architecture Model activity.

(6) Certify Software: This activity verifies that the developed software exhibits the same behavior as is defined in the specification.

Define/Refine the Engineering Artifacts

The Cleanroom Specification Volumes are defined in three forms: a Domain-level Specification, System-level Specification and Release-level Specification. The Domain-level Specification contains high-level requirements and constraints for the Domain. The System-level Specification contains the high-level system requirements and constraints. Each Release-level Specification contains lower-level requirements as pertaining to a particular release.

The Object Model (aka Logical View of the Architecture) constitutes a logical machine independent design view of the software architecture and consists of:

(1) Class category diagrams, identifying system layers.

(2) Class diagrams depicting the static view of the problem domain which consists of classes & the relationships between classes (class collaboration).

(3) Class specifications for each class that describe the responsibility of each class. Information such as attributes, operations, and relationships are captured textually in the class specification.

(4) Object scenario diagrams that depict the dynamic view of how the classes interact to perform key system mechanisms or threads.

(5) Ada PDL giving an abstract view of Object Behavior.

The Architecture Model (aka Physical View of the Architecture) defines the executable components of the system. A display hierarchy is created using Display Builder. This hierarchy captures the user interface and menu flow. The major functional components of the system are defined using UNAS Tasks. The executable program (called a SAS) consists of UNAS Tasks grouped into UNAS Processes (Equivalent to UNIX Processes) which are allocated to hardware processors. A UNAS Task "withs" in the appropriate Ada packages to carry out it's intended functionality.

The mission components are represented as Ada packages. Miscellaneous data files are created and maintained by the various RICC tools. Trade Studies identify different mappings of Classes (Ada Packages) to UNAS tasks, and associated reasons for making decisions leading to the mappings.

Define/Refine the Mapping between the Artifacts

The iterative mapping between the major artifacts is shown in Figure 9. The mapping is shown within the context of the layered framework.

This layering creates a separation of concerns between the framework layers such that:

(1) functional requirements can change, impacting the Object Model but will not necessarily perturb the Architecture Model (including the network topology); and

(2) changes, over time, in the network topology, operating systems, and COTS products won't affect the Object Model or Specification.

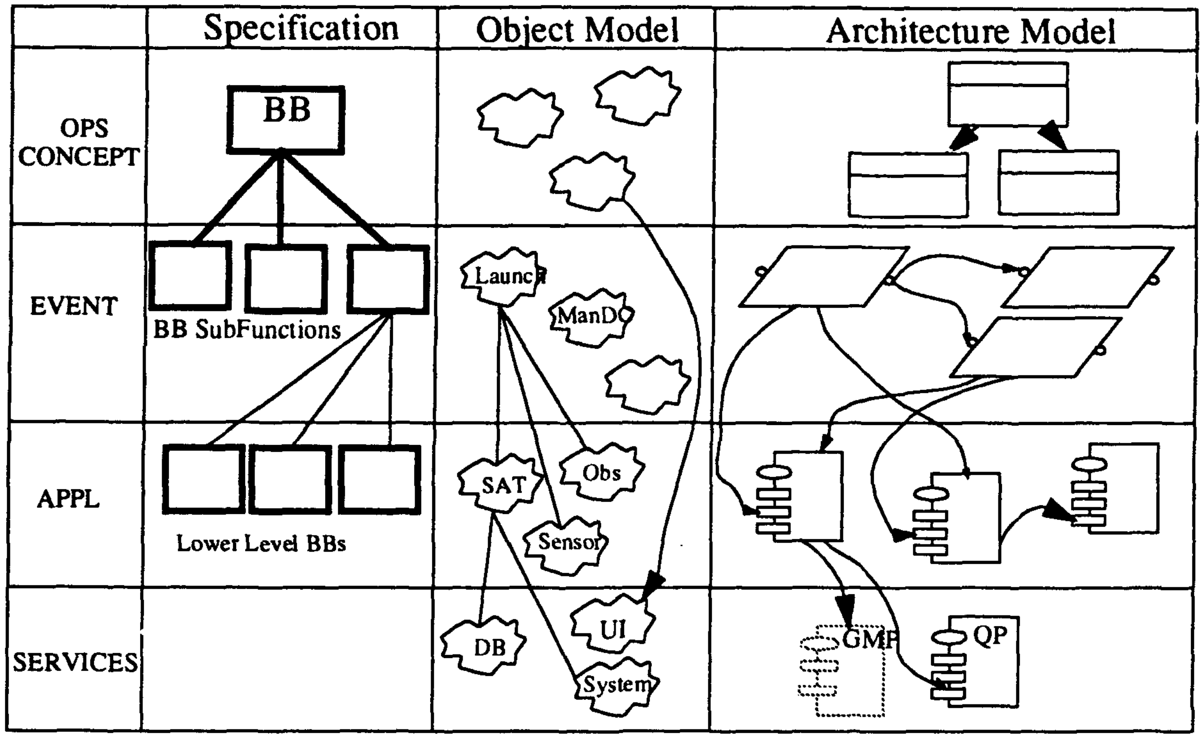


Figure 9. Artifact Mappings

Select/Adjust the Architectural Framework

A key aspect of understanding the artifacts was to define them within the context of an overall architectural framework for C^3I systems. This framework transcends the different artifacts as described below.

The Cleanroom Specification defines a set of black box subfunctions that represent the key system functionality of the system. These black box subfunctions map to the Event Layer which captures the dynamic system behavior.

The overall static object oriented architecture is represented in a layered framework depicted in Figure 9 (middle column). Each layer consists of one or more class categories. The major layers are described as:

(1) Operations Concept Layer: the user behavior derived from MOIA is captured here. The classes consist of sets of user actions (i.e. a session) that perform some functional system behavior. The dynamic interaction of user-computer is described by the use state transition diagrams.

(2) Event Layer: the dynamic system behavior that captures system policies and procedures. This is typically the most volatile part of a system. System events can be things such as satellite launches, or user procedural interactions. The classes at this layer invoke class operations at the Applications Layer to perform system functions.

(3) Applications Layer: where the tangible classes are defined. Things such as system messages are encapsulated. Usually these classes are less dynamic but message classes will encapsulate algorithmic processing. The stable, static system data is encapsulated at this level. In essence, these classes form an object layer over lower-level services such as UNAS, Query Processor & Display Builder packages.

(4) Services Layer: solution space services such as the RICC components.

The Architecture Model (aka Physical View of the application) consists of the same architectural layers as described above which are built upon the RICC components. These are:

(1) Operations Concept Layer captures the user interactions of the system. All screens and user actions are represented here. The RICC Display Builder tool creates artifacts (Ada main programs, etc.) that reside in this layer.

(2) Event Layer in which dynamic system behavior is captured in a Software Architecture Skeleton (SAS) developed using the UNAS tool. UNAS Tasks are used to capture dynamic system behavior. A system architect defines the tasks based upon the Object Model. The system architect also defines the physical layout (network topology) of the architecture. UNAS Tasks are allocated to UNAS Processes and hardware processors in this step.

(3) Application Layer which is an object layer (implemented as Ada packages) that represent information storage & retrieval while encapsulating system data. A key approach in this layer is to build an object layer on top of the Service Layer (RICC components) such that a class (its data & operations) defined by the Object Model is mapped into Ada package specs. The implementation of the class (via class specific code & calls to RICC services) is hidden in the package body.

(4) Services Layer which contains solution space services such as the RICC components.

Detail /Improve the Process

This captures the process to the level needed for practitioners who are using it to build product-line applications. Rather than attempt to define the entire process at once, an incremental approach is followed: modest portions are chosen for detailing, and the experience gained in executing the selected portions is used to guide further definition activities.

Use the Defined Process

The process definition increment concludes with its most important part: use for real project activities and feedback to help improve the process and to guide process definition activities in related areas.